ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Análisis automático de voz hablada para detección de dificultades en el aprendizaje de la lectura

Proyecto de fin de carrera presentado a la Facultad de Ingeniería de la Universidad de la República por

Gabriel De Cola y Guzmán Chalupa

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Eléctrico.

Tutores

Martín Rocamora ..... Universidad de la República
Pablo Cancela ..... Universidad de la República

Tribunal

Federico Lecumberry ..... Universidad de la República
Juan Valle Lisboa ..... Universidad de la República
Pablo Zinemanas ..... Universidad de la República

Montevideo
lunes 30 abril, 2018

Análisis automático de voz hablada para detección de dificultades en el aprendizaje de la lectura, Gabriel De Cola y Guzmán Chalupa.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 95 páginas.
Compilada el lunes 24 septiembre, 2018.
http://iie.fing.edu.uy/

Resumen

Este trabajo busca realizar la automatización del análisis de los test de nominación automatizada rápida (RAN). El test RAN es un predictor de dificultades en la lectura muy utilizado. Esta técnica requiere grabar la voz de un niño nombrando una serie de palabras que se repiten en cierta secuencia conocida, y posteriormente, escuchar y analizar la grabación generada. Este análisis posterior dificulta la utilización masiva del test RAN ya que demanda considerable tiempo de análisis. La automatización del análisis de los test RAN podría posibilitar la implementación masiva de los mismos como herramienta de screening.

El algoritmo que se implementó se basa en la idea de que el test usa elementos que se repiten a lo largo del mismo, es decir, se basa en la autosimilitud de la señal. Se implementa un detector de actividad vocal para identificar el inicio y fin de cada palabra, y se utiliza Dynamic Time Warping para comparar los segmentos de audio de voz identificados. Luego se reconstruye la secuencia de palabras utilizando Spectral Clustering, detección de outliers y distancia de edición. Finalmente se compara la secuencia obtenida con la secuencia de la mariz original del test RAN, detectando los errores cometidos.

Esta página ha sido intencionalmente dejada en blanco.

Prefacio

Resulta emocionante, para los autores de este documento, trabajar en un problema real y desafiante, como es el objeto de este proyecto. En particular, el aspecto de potencial contribución a la sociedad a través de un pequeño aporte al trabajo de quienes investigan y combaten las dificultades de aprendizaje en niños, nos inspira y nos motiva enormemente. Nos ilusiona pensar que quizás en un futuro este trabajo ayude a que mayor cantidad de niños con dificultades de aprendizaje pueda tener mejores oportunidades para superarlas y crecer de forma más plena.

Guzmán Chalupa
Gabriel De Cola

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

"La habilidad de leer y comprender textos es una de las capacidades fundamentales para la adquisición del conocimiento en las sociedades modernas. Para nuestro país tanto las evaluaciones de comprensión lectora de la ANEP como las pruebas PISA han mostrado que el nivel medio de desempeño lector es bajo en relación al mundo, muestra una gran variabilidad y no parece haber mejorado en los últimos tiempos (Consejo de educación secundaria, 2011). A diferencia del lenguaje, la capacidad para leer no se desarrolla de forma natural, pues requiere enseñanza y la práctica. Las dificultades en la lectura, en particular la dislexia del desarrollo, tiene una prevalencia de entre el 8 y 10 %. La alfabetización es un componente esencial para el desarrollo de una vida exitosa en las sociedades modernas, los niños que tempranamente encuentran dificultades para leer practican menos fuera del aula lo que genera un efecto encadenado reduciendo la adquisición de vocabulario y de conocimiento del mundo, con consecuencias sociales y psicológicas para esos niños y que deriva en serias dificultades en la vida adulta (Stanovich, 1986). La detección temprana de dificultades en la lectura es un componente indispensable para el desarrollo de intervenciones individualizadas que logren interrumpir este bucle." [23]

Este proyecto nace en respuesta a una propuesta realizada por el Centro de Investigación Básica en Psicología (CIBPsi) de Facultad de Psicología, UDELAR. En el CIBPsi se desarrollan investigaciones que buscan comprender mejor los mecanismos básicos de funcionamiento de la mente humana con un marcado enfoque experimental, a fin de desarrollar ideas innovadoras y útiles para diferentes sectores de la sociedad[1].

La propuesta del CIBPsi está vinculada a un proyecto en colaboración con el Plan Ceibal. En ese proyecto, denominado "Diseño de una evaluación digitalizada de predictores de las dificultades lectoras", se busca diseñar e implementar una batería digitalizada de tareas que permita evaluar predictores de dificultades en la lectura. Se pretende crear una interfaz lúdica para los niños y una evaluación sistemática y masiva. Finalmente se espera que los datos recopilados permitan

[1] https://cibpsi.psico.edu.uy/

predecir dificultades posibilitando intervenciones precisas y oportunas. El aporte que busca realizar este proyecto de fin de carrera es la automatización de la etapa de evaluación de un predictor de dificultades en la lectura en niños: el test RAN (Rapid Automatized Naming).

1.1. Problema: Test RAN

La nominación automatizada rápida (RAN, por sus siglas en inglés) es considerado uno de los principales predictores de desempeño lector. Típicamente consiste en la denominación secuencial de un conjunto de 50 elementos dispuestos en una matriz de 5 filas. La tarea que se plantea al niño evaluado consiste en nombrar lo más rápido posible cada elemento de la matriz. Se mide la precisión en las respuestas y el tiempo total de la tarea. Los elementos de la matriz pueden ser objetos, colores, números o letras. En particular, en la aplicación llevada a cabo por el CIBPsi en escuelas del Uruguay, se presenta una matriz de 5 x 6 con 6 repeticiones de 5 elementos (ver Figura 1.1). El niño debe nombrar en voz alta cada elemento (e.g. "azul", "rojo", "blanco"), siguiendo un orden análogo al de la lectura, de izquierda a derecha y de arriba a abajo. La aplicación típica de estos tests implica que una persona debe presentar la tarea al niño, y grabar la realización en un archivo de audio. En una etapa posterior, una persona debe analizar el audio y medir el desempeño a través de las respuestas las respuestas, detectando posibles discrepancias con el orden correcto de lectura del test.

Figura 1.1: Ejemplos de matrices de test RAN

La aplicación masiva de este tipo de test con la metodología empleada hoy en día por el CIBPsi, se hace dificultosa debido a la cantidad de horas hombre que demanda. En particular, el análisis posterior de las grabaciones es una tarea que requiere mucho tiempo.

El objetivo de este proyecto se centra en la automatización de la etapa de análisis de la grabación producto de la aplicación de un test RAN. Se busca implementar un prototipo que realice esta tarea. Eventualmente, aunque fuera del alcance de este proyecto, se considera la posibilidad de integrar esta herramienta a las computadoras XO del Plan Ceibal, también conocidas como "ceibalitas". Esto último se realizaría en el marco de una interfaz lúdica para los niños, implementando una evaluación sistemática, masiva y semiautomatizada.

1.2. Solución propuesta

El algoritmo que se propone implementar se basa en la idea de que en el test los elementos de interés se repiten a lo largo del mismo, es decir, se basa en la autosimilitud de la señal. Por ejemplo, en la matriz de la derecha en la figura 1.1, el elemento "gato" aparece un total de seis veces, y por tanto en el audio también debería repetirse la palabra "gato", la misma cantidad de veces y en el mismo orden. Por lo tanto, buscamos separar cada palabra en la grabación y determinar si dos palabras cualquiera son iguales o diferentes. Luego quedará establecida la posición en que se repiten las palabras correspondientes a cada elemento de la matriz, y así se determinará si el patrón en la matriz y en el audio analizado son iguales o no.

Notar que este enfoque para resolver el problema es independiente del lenguaje o del idioma, como se ilustra en la Figura 1.2. El algoritmo busca reconocer estructuras iguales entre sí y diferentes del resto, sin importar la forma específica de esas estructuras. Siempre y cuando los elementos de la matriz se lean empleando palabras distintas entre sí, el sistema debería encontrar la posición en que diferentes instancias de las mismas palabras se repiten, y dónde las palabras son distintas entre sí.

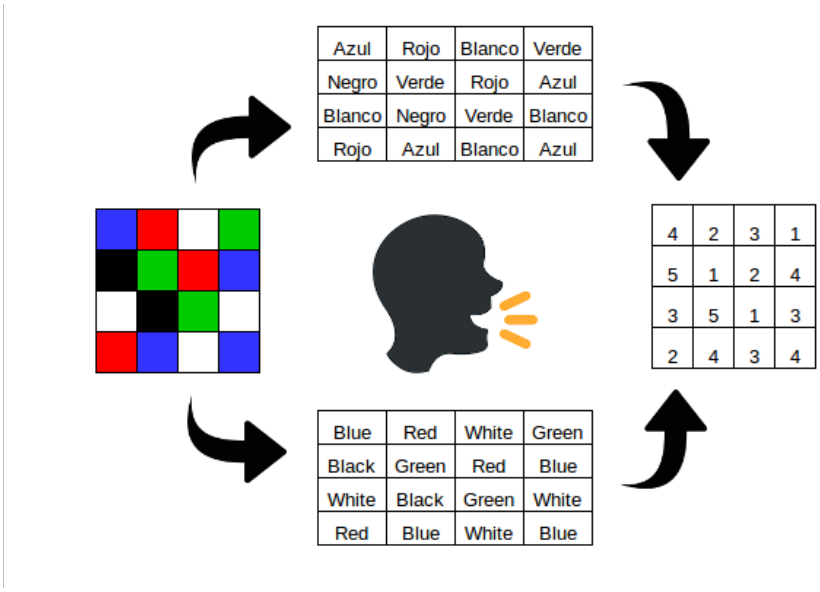


Figura 1.2: El algoritmo propuesto es independiente del lenguaje

Esta independencia del lenguaje, permite que se puedan evaluar instancias de tests RAN que empleen matrices con objetos diferentes, por ejemplo donde se cambie el color verde por el color amarillo, que cada color se cambie por una letra específica, etc.

Notar que esta estrategia para tratar el problema presupone implícitamente que en la señal de audio a analizar, toda palabra articulada presente corresponde

a la persona evaluada y a la lectura de los elementos de la matriz de test. Esto es así porque el sistema toma las palabras identificadas en la segmentación y las compara a todas entre sí, y no se realiza una identificación del hablante, lo cual agregaría otro nivel de complejidad al problema.

Las hipótesis sobre las grabaciones de audio a analizar son:

- Las grabaciones corresponden a voz humana de una sola persona, en ausencia de otras conversaciones.
- Las palabras registradas en la grabación corresponden únicamente a los elementos leídos del test RAN, admitiendo posibles outliers, es decir palabras que no forman parte del test. Estos outliers deben ser pocos (se admiten 3 o menos, en caso de ser la misma palabra repetida).
- Existe una pausa, o un silencio, entre palabras.
- Las palabras pronunciadas son articuladas correctamente. Esto corre en particular para el test de letras donde, por ejemplo, para el elemento "M", que debería leerse "eme", la articulación "mmm" se considera incorrecta.
- Buena relación señal a ruido.

Se decidió usar el lenguaje de programación Python para desarrollar el sistema. Esta decisión fue basada en que, por un lado, este lenguaje es relativamente sencillo y simple, aunque poderoso, y existen muchas bibliotecas disponibles que facilitan mucho la resolución de los problemas. Por otro lado, en las computadoras del plan Ceibal normalmente se utiliza Python para este tipo de aplicaciones, y esta elección facilitaría la posible integración del sistema a esta plataforma.

Para la implementación de la solución propuesta, se propone un sistema que tome dos entradas y devuelva una salida, como se muestra en la figura 1.3.

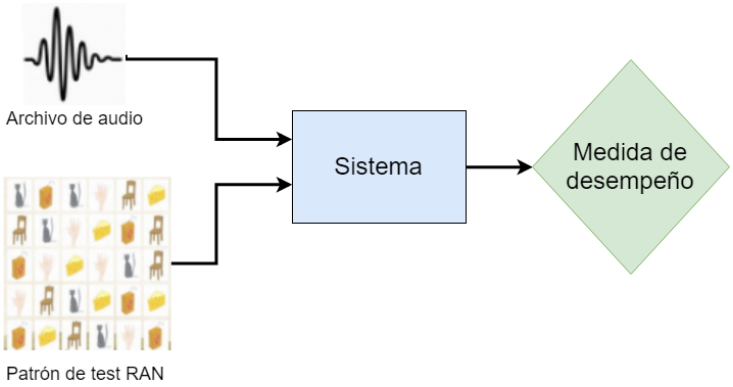


Figura 1.3: Diagrama de entradas y salidas del sistema propuesto

ENTRADAS:

1. El archivo de audio con la grabación de un test RAN.
2. La secuencia original de elementos en el test RAN.

SALIDAS:

1. Un valor que cuantifique el desempeño del niño en su realización del test RAN.

Observar que para la evaluación del test RAN como predictor de dificultades de lectura, el dato de la duración de la tarea también es importante. Dado que proveer esta información a partir de un archivo de audio digital es trivial, no lo consideramos como una salida del sistema diseñado.

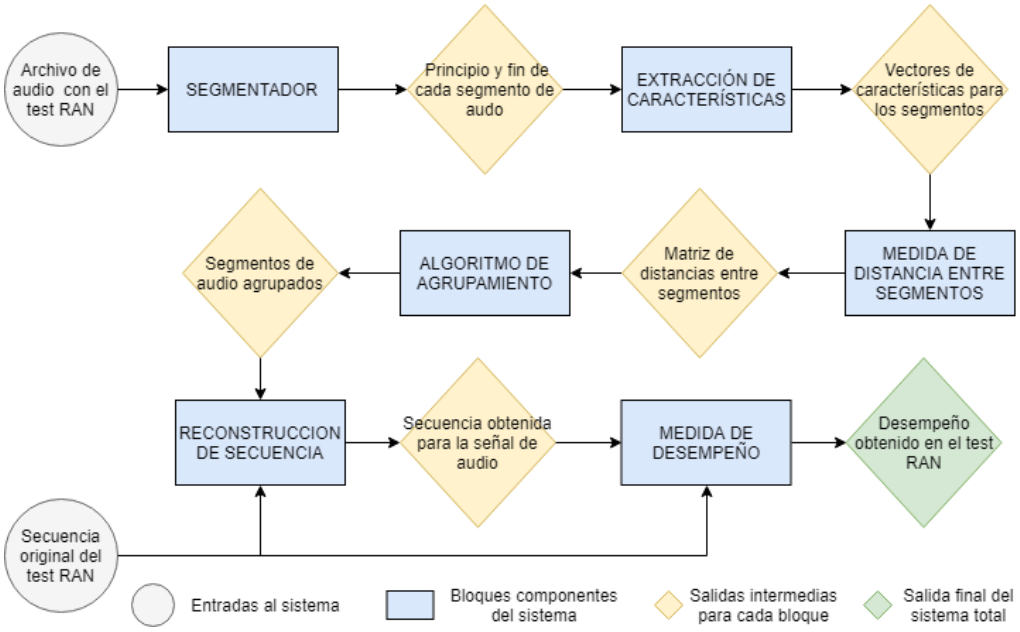


Figura 1.4: Diagrama del sistema

Como se puede ver en el diagrama del sistema de la Figura 1.4, la primer etapa de procesamiento consiste en un segmentador de palabras automático. La función de este módulo es entregar a su salida el instante de tiempo en que comienza y termina cada una de las palabras presentes en la grabación. Esto se logra discriminando instantes de tiempo en que existe actividad de voz, de instantes de tiempo de silencio. Se probaron dos algoritmos diferentes para realizar esta tarea y se eligió el que brindó mejores resultados. Para esto fue necesario definir una forma de evaluación del desempeño del segmentador.

Luego de la segmentación, para cada palabra se realiza una representación en vectores de características, preparando a la señal para la etapa siguiente. Este proceso comienza dividiendo la señal correspondiente a cada palabra en fragmentos de audio pequeños, a los que se denomina *frames* o tramas, con cierto solapamiento

temporal entre sí. Luego se calcula para cada trama, los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC[2]).

El paso siguiente es comparar cada fragmento de audio identificado como una palabra en la señal original, con el resto de las palabras halladas, y decidir cuándo dos palabras son la misma, y cuándo son diferentes. Es preciso, para este fin, definir una medida de distancia entre palabras, que devuelva un valor pequeño para palabras iguales y un valor grande para palabras diferentes. Se implementó Dynamic Time Warping (DTW[3]) como solución a este subproblema. DTW a su vez, requiere utilizar algún tipo de medida o distorsión entre vectores de características, es decir una medida entre tramas de audio. Para esto último se eligió la medida de distancia cepstral ponderada por la función seno elevado.

Una vez calculada la distancia entre cada palabra con todas las restantes, se obtiene una matriz con esta información. Luego hace falta decidir para qué valores de distancia se considera que dos palabras son iguales y cuándo distintas; e identificar la secuencia en que se repiten las palabras. Esto lo realiza un algoritmo de agrupamiento de Spectral Clustering[4]. La salida de este módulo es una secuencia ordenada de elementos que se repiten, imitando a la secuencia hablada original.

Finalmente solo queda evaluar qué tan parecida es la secuencia obtenida a la secuencia de referencia propuesta por el test RAN. Para esto es necesario que el sistema conozca el patrón con que se repiten los elementos presentes en la matriz que el niño leyó en el momento de aplicación del test. Esta evaluación se realiza utilizando una medida de distancia de edición de texto.

$$d_c^2(x, y) = \sum_{n=-\infty}^{\infty} (c_n^x - c_n^y)^2$$

1.3. Antecedentes

Existe variedad de trabajos que buscan detectar dificultades en el lenguaje por medio del procesamiento digital de señales de voz. Por ejemplo, en [2], se mencionan decenas de herramientas que estudian diferentes aspectos de la voz, para detectar distintas patologías.

Lo que a este trabajo concierne, en particular, es la evaluación automática del test RAN. Esta tarea es realizada en el sistema desarrollado por Jyrki Rissanen en 2008 llamado Naming Game [18].

Se trata de un juego que presenta al usuario con una matriz con dibujos que representan objetos. El programa destaca las líneas en orden desde la primera fila hacia abajo, a medida que la persona que toma el test va nombrando en voz alta cada uno de los objetos. El programa reconoce las articulaciones de palabras en tiempo real. Esto permite que se muestre al niño testeado una fila de la matriz a

[2]Se desarrolla en el capítulo 3
[3]Se desarrolla en el capítulo 4
[4]Se desarrolla en el capítulo 5

la vez, ya que, el sistema lleva el registro del avance a medida que las palabras son leídas en voz alta. De esta forma al completar una línea, se pasa a la siguiente.

El programa posee algunas herramientas interesantes. Entre ellas, permite al investigador examinar la grabación registrada en el momento del test y acceder a reportes estadísticos con información como velocidad promedio al nombrar ciertos objetos, desviación estándar, etc.

Este sistema utiliza un algoritmo de reconocimiento de voz simple que reconoce palabras analizando solamente el patrón de energía registrado en la señal de audio. Si el patrón de energía de una palabra pronunciada, normalizado por su duración y niveles de energía, es suficientemente cercano a alguna de las palabras esperadas, entonces el sistema reconoce la palabra. Esto implica que es necesario tener registro previo de patrones de energía para cada palabra, y comprobar que el sistema puede diferenciar correctamente todas las palabras en su registro. Es importante destacar que este tipo de algoritmo solamente puede diferenciar palabras significativamente distintas una de otra.

1.4. Base de Datos

Para poder evaluar el desempeño del sistema el CIBPsi nos facilitó acceso a grabaciones realizadas al aplicar los tests a niños en escuelas de Uruguay. Estas grabaciones resultaron no ser adecuadas para evaluar el sistema ya que no se habían realizado teniendo en cuenta las restricciones que impone esta implementación de evaluación automática: En estas grabaciones se presentan otros hablantes además del evaluado, y a su vez, ocurren interrupciones de otros interlocutores, ruidos fuertes, etc.

Se hizo necesario entonces generar grabaciones realizadas en ambientes controlados y sin intervenciones externas. En una primer instancia, se grabaron doce audios, dos realizaciones del test RAN, más dos para cada una de las cinco palabras mencionadas, donde se repite una misma palabra varias veces (aproximadamente 50 instancias para cada una) y con distintas posibles cadencias o entonaciones. Estas grabaciones permitieron hacer los primeros acercamientos, pero luego se hizo necesario generar una base de datos para realizar un entrenamiento para definir algunos parámetros clave del sistema. Estas grabaciones debían estar etiquetadas con la información de principio y fin de cada palabra.

Con la colaboración de 20 hablantes, se grabaron 120 simulaciones de un test RAN de cinco palabras (”gato”, ”jugo”, ”mano”, ”queso” y ”silla”) incluyendo varios casos con posibles errores:

1. Sin errores
2. Agregando la repetición consecutiva de algunos de los elementos de la secuencia original.
3. Agregando a la secuencia original elementos que no pertenecen al test, por ejemplo ratón”.

4. Eliminando elementos de la secuencia original

5. Agregando elementos que no pertenecen al test y eliminando elementos de la secuencia original.

En el apéndice A se detallan las características de la base de datos.

1.5. Entrenamiento y Evaluación

El sistema propuesto tiene algunos parámetros cuyos valores pueden hacer variar significativamente el desempeño del sistema, y si bien hay restricciones o rangos sugeridos para los mismos, es difícil escoger y fundamentar el uso de valores específicos. Los valores de estos parámetros se definen mediante experimentos.

Luego de implementar y comenzar a probar el segmentador de palabras, se vio que este introducía gran cantidad de errores al sistema. Esto determinó que se decidiera evaluar el sistema en dos etapas bien diferenciadas: el segmentador por un lado, y la comparación de palabras y agrupamiento por otro lado. Para cada uno de los dos subsistemas se definió una medida de desempeño, y se realizó entrenamiento y prueba de los parámetros clave utilizando el método de Validación Cruzada. A su vez, luego de hallar los parámetros que mejor se ajustaban a la base de datos generada para hacer los experimentos, se realizó una prueba de concepto. En esta prueba se probó el desempeño del sistema con grabaciones de audio tomadas de realizaciones de test RAN en niños de escuelas de Uruguay.

1.6. Organización

La organización de los siguientes capítulos se detalla a continuación.

En el capítulo 2, se establece qué es un segmentador de palabras y su importancia en el sistema. Se comentan los dos algoritmos estudiados, se los compara y se muestra la medida de desempeño obtenida, permitiendo elegir la implementación con mejor performance.

Luego, en el capítulo 3, se ve brevemente un modelo del aparato fonador humano y cómo existe una fuerte correlación entre los llamados Coeficientes Cepstrales en las Frecuencias de Mel (MFCC) y la articulación del aparato fonador para los distintos sonidos generados al hablar. Esta relación es la que permite representar palabras como secuencias de vectores de características para poder realizar el análisis de la señal.

Al llegar al capítulo 4, se estudia el algoritmo de Dynamic Time Warping (DTW), utilizado en el sistema para comparar palabras. Se define una forma de implementación del mismo, y cómo se trabajó para optimizar la función desarrollada.

En el capítulo 5 se explica la función del bloque de agrupamiento, y se estudia la técnica utilizada, Spectral Clustering.

En el capítulo 6, se define la medida de desempeño del sistema, se detallan las pruebas realizadas para entrenar y evaluar el sistema, y se muestran los resultados obtenidos.

Finalmente en el capítulo 7, se realizan algunas observaciones a la vista del sistema desarrollado y el desempeño del mismo.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Segmentador de palabras

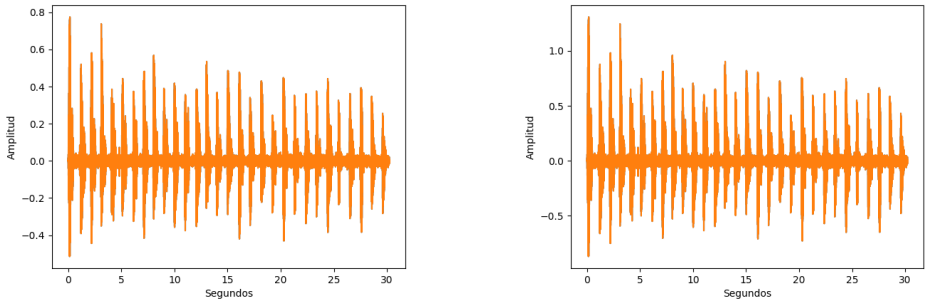
2.1. Segmentador Automático de Palabras

El sistema de procesamiento de audio que se plantea en este proyecto incluye un segmentador automático de palabras como primera etapa del procesamiento. La segmentación automática de palabras es el proceso de identificar, en una señal de audio con habla, el instante de tiempo de comienzo y el instante de tiempo de fin de cada palabra de forma automática.

El segmentador de palabras está basado en el uso de un detector de presencia de habla, VAD por sus siglas en inglés (Voice Activity Detector). Tiene muchas aplicaciones en codificación y reconocimiento de voz.

El algoritmo típico de un VAD comienza con una etapa de filtrado o reducción de ruido. Luego se extraen características que describen a la señal de audio a analizar. Las características son usadas para clasificar una sección de la señal en una de las dos siguientes clases: 1) Presencia de habla y 2) Ausencia de habla. Generalmente, este algoritmo depende de un umbral que separa a las dos clases [10].

Se consideraron dos algoritmos de VAD para realizar la segmentación. Se decidió no aplicar un filtrado o reducción de ruido a las señales de audio previo a procesarlas con el segmentador. En su lugar, la reducción de ruido se realiza internamente en el algoritmo de la segmentación, como se explicará más adelante. En ambos casos se aplica previamente una normalización de la ganancia. El factor de normalización se elige encontrando el máximo de la señal resultante de aplicar un filtro de mediana a la señal de audio en el dominio del tiempo. El filtro de mediana busca ajustar las posibles muestras alteradas por ruido en la señal cuya magnitud sea considerablemente mayor que sus muestras adyacentes. Así se asegura que el factor de normalización no se elija a partir de una muestra ruidosa. Este filtro solamente se usa para calcular el factor de normalización y no es aplicado a la señal con la que se trabaja. Decidimos fijar el largo del filtro en 9 muestras ya que se encuentra en el centro del rango de valores (entre 7 y 11 muestras) para el que se observaron buenos resultados. En la figura 2.1 se observar como cambia una señal de audio después de la normalización mediante este método.



(a) Señal sin normalizar (b) Señal normalizada

Figura 2.1: Normalización de una señal de audio

Es conocido que el rango de frecuencias para el cual se concentra la mayoría de la energía de una señal de voz hablada se extiende, aproximadamente, de 300 Hz a 3400 Hz, como se puede observar en la Figura (2.2). En este documento se referirá a esta banda como la banda vocal.

El primer segmentador implementado (llamémoslo A) se basa en un algoritmo que calcula la razón entre la energía de la señal en la banda vocal y la energía en la banda total de frecuencias cuyo rango es de 0 Hz a $F_s/2$, donde F_s es la frecuencia de muestreo de la señal [17]. Si para una sección la razón supera un umbral impuesto, elegido experimentalmente, el segmentador clasifica a la sección como presencia de habla o, en el caso contrario, como ausencia.

El segmentador A, luego de la normalización, procede separando el audio en ventanas de largo de 30 ms. Los centros de las ventanas están espaciados cada 15 ms, por lo tanto, dos ventanas adyacentes tienen un solapamiento del 50 %. Para cada ventana se calcula su Transformada de Fourier Discreta (DFT)[1]. La DFT de una señal está compuesta por muestras espaciadas que corresponden a valores de frecuencia discretos o *bins* de frecuencia. Los *bins* están espaciados cada F_s/N, donde N es el número de muestras de la señal. Para cada *bin* (b) se halla su energía respectiva. Luego se suma la energía de todos los *bins* en el rango de frecuencias de la banda vocal y la energía de todos los *bins* en la banda total y se hace el cociente (C) entre los valores obtenidos:

$$C = \frac{\sum\limits_{f\epsilon[300Hz,3400Hz]} X^2(f)}{\sum\limits_{f\epsilon[0Hz,F_s/2Hz]} X^2(f)} \text{ donde } X = FT\{x\}$$

Si el valor de C es mayor o igual al umbral impuesto (u_A) se le asigna a la ventana correspondiente un valor "Positivo", elegido como 1, y en el caso contrario

[1]Descripción en Apéndice D

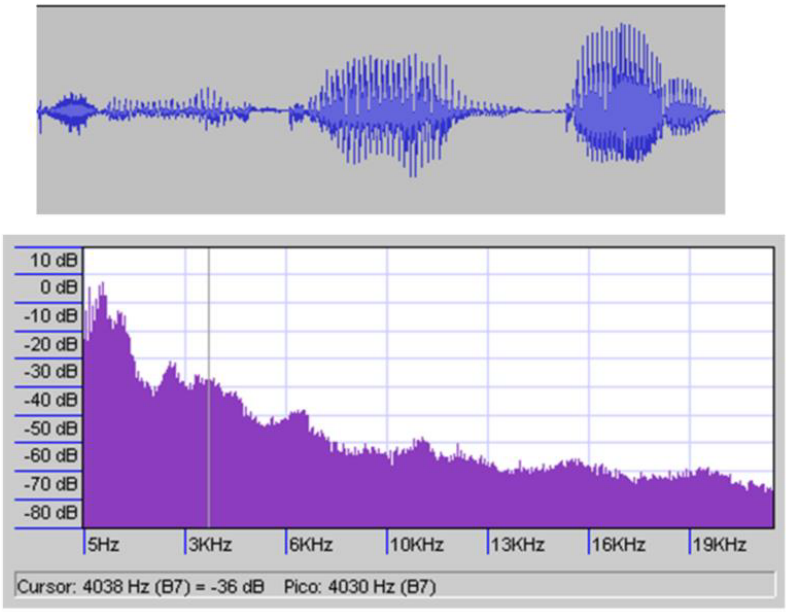


Figura 2.2: Espectro típico de la voz (Imágen sacada de [7])

se le asigna "Negativo", elegido como 0. El resultado es una secuencia compuesta de unos y ceros de largo igual a la cantidad de ventanas definidas para la señal de audio. La secuencia va a ser procesada con intención de mejorar la segmentación y se va a explicar luego de desarollar la implementación del otro segmentador ya que el procedimiento es idéntico para ambos.

El segundo segmentador B usa como criterio la Energía en Tiempo Corto (STE), la cual es una característica muy utilizada para la clasificación de sonido y silencio en una señal de audio. La STE se define como:

$$E_n = \sum_{m=-\infty}^{+\infty} (x[m]w_1[n-m])^2$$

Donde E_n es la energía correspondiente a una ventana de audio y w_1 es la ventana de suavizado, que puede ser de varios tipos, tales como Rectangular, Hann, Hamming, entre otros. Las ventanas de análisis, las cuales se tomaron rectangulares p tienen un largo de 30 ms y sus centros están espaciados cada 15 ms.

Se probó utilizar también la Taza de Cruces por Cero (Zero Crossing Rate en inglés o ZCR) como característica discriminante de no habla y habla, ya que es un indicador de la presencia o ausencia de componentes de alta frecuencia en la señal de audio. La presencia de altas frecuencias está más correlacionada con el sonido producido por consonantes que con el silencio a causa de la ausencia del habla. Finalmente, en las pruebas realizadas resultó ser un factor bastante variable para cada audio y se decidió descartarlo y trabajar solo con la STE.

Se considera que hay silencio en una señal de audio cuando la STE es inferior a un umbral (u_B). Al igual que el segmentador A, el umbral se determina experimentalmente y después de comparar la STE de cada ventana de audio con el umbral se tiene una secuencia de unos y ceros representando los casos de sonido y silencio respectivamente.

Para mejorar el resultado de los segmentadores implementados se aplicaron ciertas etapas de procesamiento posterior. Con esto se pretende juntar segmentos de audio independientes obtenidos a partir de la segmentación que en realidad, son parte de una misma palabra. Este problema se presenta por los silencios entre sílabas de una misma palabra. Los espacios clasificados como silencio se pueden deber a la mecánica del habla para ciertas palabras, ya que algunas llevan un silencio entre sus sílabas conformantes, o incluso por la forma de hablar de la persona en específico. Además busca corregir casos de confusión entre consonantes y silencio que se pueden dar debido al bajo contenido de energía que conllevan ciertas consonantes, principalmente las mudas.

El primero de los procesos usados consiste en remplazar sub-secuencias aisladas, o sea, que si en la secuencia hay ceros o unos aislados estos serán remplazados por su contraparte. Se consideró que una sub-secuencia es aislada cuando tiene largo de dos ventanas o menor. En una segunda etapa, se busca si dos sub-secuencias de positivos se encuentran separadas por una sub-secuencia de negativos corta, entonces a la sub-secuencia Negativa se la transforma en Positiva y las tres sub-secuencias mencionadas pasan a ser una única sub-secuencia Positiva. Para esto, se considera un máximo espacio entre sub-secuencias positivas (que llamaremos *gap*). El valor para *gap* se deduce en experimentos que se explican en la sección 2.2. Para terminar, se incluyó un largo mínimo de palabra como última etapa de procesamiento posterior. Éste se fijó en 0,3 segundos ya que las duraciones de todas las palabras en los audios estaban por encima de este umbral. Este criterio cumple la función de eliminación de de ruido ya que si se detecta un sonido más corto que lo estimado seguramente corresponda a un ruido externo no deseado.

En la Figura 2.3 se observa los resultados de aplicar el algoritmo del segmentador B a una señal de audio conteniendo 4 instancias de palabras y en la Figura 2.4 se ve lo mismo pero sobre la STE de esta señal. La señal de audio utilizada en el ejemplo es el comienzo de una grabación más larga de un test RAN, los cuales son el foco principal del proyecto.

2.2. Entrenamiento y validación del segmentador automático

Es necesario definir una manera de evaluar el segmentador automático. Primero para aclarar, definimos una etiqueta como el par de instantes de principio y fin para una palabra. Entonces, se decidió como forma de evaluación considerar cada una de las etiquetas reales ubicadas en la base de datos (en inglés *ground truth*) e intentar encontrar si existe alguna etiqueta similiar de los fragmentos de audio

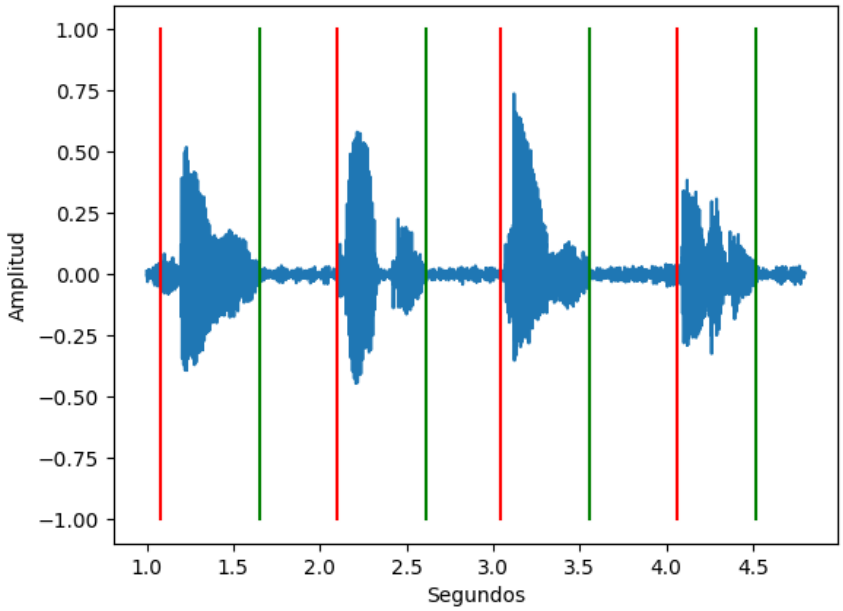


Figura 2.3: Segmentación de una grabación donde se dicen 4 palabras (comienzo:rojo, fin:verde)

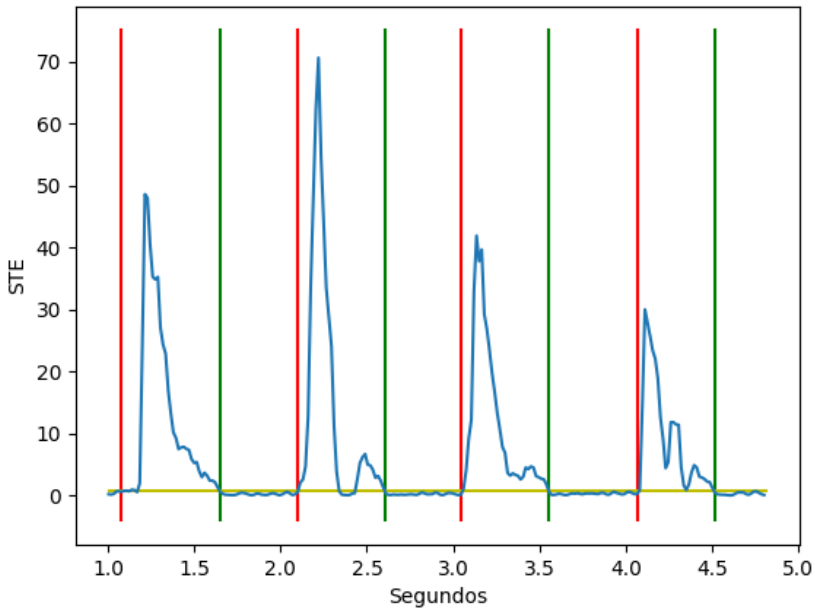


Figura 2.4: Idem a la Figura 2.3 pero se grafica en vez la STE de la señal de audio

proporcionados por el segmentador. En el caso de que se encuentre una etiqueta similar se considera que la palabra fue correctamente segmentada y en el caso contario no. Para definir si dos etiquetas son similares se evalúa si la diferencia de los principios y la diferencia de los fines de ambas etiquetas es menor a un umbral elegido. Éste se fijó en 0,2 s considerando que no puede ser muy grande ya que se pierde validez en la evaluación y que no puede ser muy chico debido a que el etiquetado real es hecho a mano. Este método tiene el problema de que podría haber casos de etiquetas proporcionadas por el segmentador que no corresponden a ninguna palabra real en la señal de audio, por ejemplo, un ruido externo. Sin embargo, en el sistema total planteado se eliminan palabras que no son similares a otras palabras del mismo archivo de audio (ver sección Sección 5.2). Esto permite en cierta forma sortear este tipo de errores en el segmentador.

En definitiva se define el desempeño del segmentador automático (DES_{SA}) para una grabación en particular como:

$$DES_{SA} = \frac{N_{ec}}{N_{gt}},$$

donde N_{ec} es igual a la cantidad de palabras etiquetadas correctamente y N_{gt} es la cantidad de palabras reales en la grabación.

Una vez definido esto, se pasaría a entrenar el segmentador, o sea, encontrar los parámetros óptimos del mismo. Para esto normalmente se haría una división del conjunto de datos en un subconjunto de entrenamiento y un subconjunto de prueba. Debido a que nuestra base de datos no es muy numerosa en datos se decidió utilizar validación cruzada (VC)[2] para validar al modelo, ya que esto permite independizar la elección de conjunto de entrenamiento y conjunto de prueba. Al tener exactamente 20 hablantes se decidió hacer la división de datos cada 2 hablantes. La intención primaria era realizar la división cada un hablante pero debido a los períodos largos de tiempo que lleva realizar las pruebas se realizó la VC con 18 hablantes para el conjunto de entrenamiento y 2 para el conjunto prueba, dando lugar a 10 grupos en la VC.

Para el segmentador A, el entrenamiento fue hecho usando el siguiente rango de valores para los parámetros u_A y *gap*:

$$u_A \in [0{,}3\ 0{,}6] \text{ cada } 0{,}02$$

$$gap \in [90\text{ ms}\ 180\text{ ms}] \text{ cada } 15\text{ ms}$$

La elección del rango para *gap* se realizó así debido a que las ventanas fueron construidas con 15 ms de separación, por lo tanto, se tiene que trabajar con múltiplos de ese valor. El rango para STE simplemente fue elegido empíricamente en base a experimentos exploratorios iniciales.

Para el entrenamiento del segmentador B, se utilizó el mismo rango para *gap* y para el umbral u_B se utilizó el siguiente rango de valores:

[2]Descripción en Apéndice F

$$u_B \in [0{,}6\ 0{,}9]\ \text{ cada }\ 0{,}02$$

Se termina definiendo al desempeño del segmentador para la base de datos como la media aritmética de todos los desempeños obtenidos en cada una de las iteraciones de la validación cruzada. Para el segmentador B se obtuvo un desempeño promedio de 92,87 % y para el A se obtuvo un desempeño de 81,19 %. Ya que el segmentador B tuvo un mejor desempeño se optó por trabajar con éste y la discusión se enfocará en el mismo. En las tablas 2.1 y 2.2 se muestran los resultados obtenidos para el segmentador B. Los valores óptimos para el segmentador B en el entrenamiento usando validación cruzada fueron siempre los mismos:

$$STE_{optimo} = 0{,}7$$

$$gap_{optimo} = 150\text{ms}$$

Considerando que este es el caso, se puede asumir que estos serían los parámetros óptimos si se usara toda la base de datos para el entrenamiento. En la Figura 2.5 se puede observar como varía el desempeño promedio con los dos parámetros mencionados. El punto óptimo encontrado se marca con una línea vertical negra.

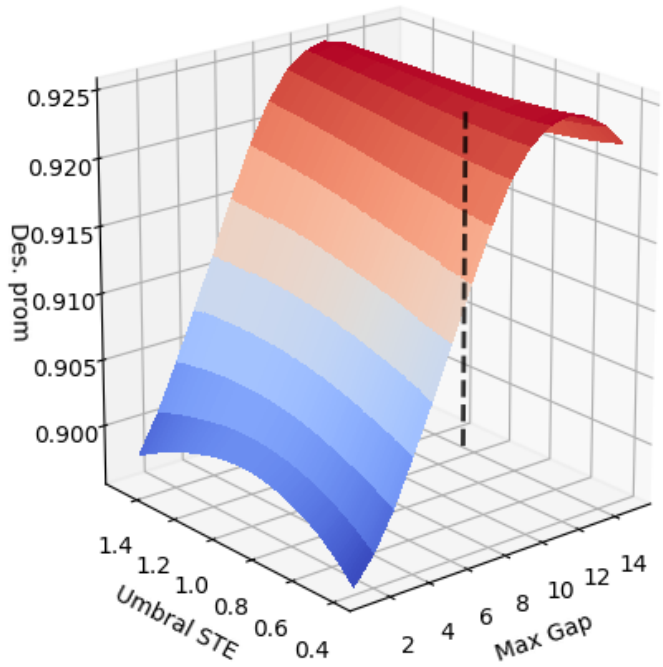


Figura 2.5: Desempeño promedio variando u_B y gap

GRUPO CV	DESEMPEÑO ENTRENAMIENTO	DESEMPEÑO PRUEBA
1	93.3 %	91.3 %
2	92.8 %	97.9 %
3	93.1 %	94.9 %
4	92.7 %	97.8 %
5	94.5 %	82.7 %
6	94.0 %	87.4 %
7	92.8 %	96.9 %
8	92.8 %	94.9 %
9	93.4 %	90.9 %
10	93.1 %	94.0 %

Tabla 2.1: Resultados para la Validación Cruzada en el entrenamiento del segmentador automático

En la tabla 2.1 se observa el máximo desempeño en el entrenamiento para cada grupo de la VC, el cual se obtiene utilizando los valores óptimos de STE y *gap*. Junto a cada uno de estos valores se encuentra el desempeño de la partición de prueba respectivo al grupo de la VC. El mínimo valor para el desempeño de prueba es 82,7 % para el grupo 5 y el máximo es 97,9 % para el grupo 2. En la tabla 2.2 se puede ver el desempeño para cada audio y se observa que el problema en el grupo 5 radica en el hablante 08, el cual pertenece al conjunto de prueba para ese grupo. Analizando los audios pertenecientes a ese hablante, se observa que el principal problema es que el hablante deja menos espacio de silencio entre cada palabra, en comparación a los otros hablantes. Esto puede generar que el segmentador junte palabras adyacentes y las etiquete como una sola. En general se nota que para los hablantes rápidos se tiene peor desempeño en todos los audios mientras que en los hablantes con más pausas entre palabras la segmentación es buena e incluso en algunos casos perfecta.

2.3. Decisiones respecto al segmentador automático

Se decidió utilizar el segmentador B ya que se obtuvo un mejor desempeño con éste en la validación cruzada, que con el segmentador A(ver 2.2). Además, ya se habían realizado experimentos exploratorios previos que mostraban que el segmentador A fallaba a menudo en la detección de las consonantes s y j mientras que el segmentador B no tenía problemas con ninguna consonante en particular. Esto se puede deber a que estas consonantes poseen componentes de peso en altas frecuencias fuera del rango vocal definido. También se exploró la posibilidad de aumentar el umbral para solucionar el problema con la detección de estas consonantes pero aparecían fallas por otros lados. Las fallas incluían corrimiento no deseado entre los tiempos de principio y tiempos de fin de la palabra e incluso en algunas ocasiones dos palabras eran etiquetadas como una sola ya que el silencio entre las mismas era detectado como sonoro. Nunca se logró en estos experimentos encontrar un umbral que funcione en general para todos las palabras.

	Lista 1	Lista 2	Lista 3	Lista 4	Lista 5	Lista 6
H00	100 %	76.5 %	85.7 %	85.7 %	86.7 %	80.0 %
H01	100 %	91.2 %	100 %	92.9 %	100 %	96.7 %
H02	100 %	97.1 %	94.3 %	100 %	96.7 %	96.7 %
H03	96.7 %	97.1 %	100 %	100 %	96.7 %	100 %
H04	60.0 %	82.4 %	100 %	100 %	96.7 %	100 %
H05	100 %	100 %	100 %	100 %	100 %	100 %
H06	100 %	100 %	100 %	100 %	96.7 %	100 %
H07	100 %	100 %	100 %	96.4 %	96.7 %	83.3 %
H08	83.3 %	64.7 %	65.5 %	71.4 %	53.3 %	53.3 %
H09	100 %	100 %	100 %	100 %	100 %	100 %
H10	93.3 %	64.7 %	91.4 %	92.9 %	80.0 %	86.7 %
H11	100 %	91.2 %	77.1 %	78.6 %	93.3 %	100 %
H12	96.7 %	85.3 %	100 %	92.9 %	100 %	100 %
H13	96.7 %	94.1 %	100 %	100 %	86.7 %	100 %
H14	96.7 %	94.1 %	100 %	100 %	100 %	93.3 %
H15	100 %	97.1 %	74.3 %	100 %	100 %	83.3 %
H16	100 %	94.1 %	100 %	100 %	96.7 %	100 %
H17	83.3 %	88.2 %	94.3 %	64.3 %	90.0 %	80.0 %
H18	100 %	97.1 %	91.4 %	96.4 %	93.3 %	100 %
H19	100 %	94.1 %	85.7 %	96.4 %	93.3 %	80.0 %

Tabla 2.2: Desempeño del segmentador automático para cada audio descritos por hablante (H) y lista

También se decidió dividir el sistema total en dos módulos: uno es el segmentador automático y el otro es el resto del sistema total, que se denominará como "clasificador". La motivación viene de analizar cada módulo independientemente de los resultados obtenidos externos al mismo, reduciendo así la dificultad del problema completo. Para la evaluación del clasificador se trabajó utilizando las etiquetas de las palabras segmentadas manualmente, las cuales forman parte de la base de datos.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Extracción de características

En este capítulo se describe cómo se transforma una señal de audio en una representación apropiada para realizar el análisis deseado de la información de sonido. La extracción de características es un punto indispensable en la mayoría de los sistemas de reconocimiento de voz.

3.1. Mecanismo Vocal

La voz humana se produce voluntariamente por medio del aparato fonatorio. Éste está formado por los pulmones como fuente de energía en la forma de un flujo de aire, la laringe, que contiene las cuerdas vocales, la faringe, las cavidades oral (o bucal) y nasal y una serie de elementos articulatorios: los labios, los dientes, el alvéolo, el paladar, el velo del paladar y la lengua [12] (Figura 3.1).

La abertura entre ambas cuerdas vocales se denomina glotis. Cuando las cuerdas vocales se encuentran separadas, el aire pasa libremente y prácticamente no se produce sonido. Es el caso de la respiración.

Cuando la glotis comienza a cerrarse, el aire que la atraviesa proveniente de los pulmones experimenta una turbulencia, emitiéndose un ruido de origen aerodinámico conocido como aspiración (aunque en realidad acompaña a una espiración o exhalación). Esto sucede en los sonidos denominados "aspirados" (como la h inglesa).

Al cerrarse más, las cuerdas vocales comienzan a vibrar produciéndose un sonido tonal, es decir periódico. La frecuencia de este sonido depende de varios factores, entre otros del tamaño y la masa de las cuerdas vocales, de la tensión que se les aplique y de la velocidad del flujo del aire proveniente de los pulmones. Tanto al aumentar la tensión como al incrementarse la velocidad del flujo de aire, la frecuencia crece, produciendo sonidos más agudos.

Es posible cerrar totalmente la glotis. En este caso no se producen sonidos. Esto sucede por ejemplo durante la deglución.

Las cavidades faríngea, oral y nasal junto con los elementos articulatorios modifican los sonidos producidos por las cuerdas vocales, agregan partes distintivas a los mismos, e inclusive producen sonidos propios. Todo esto se efectúa por dos

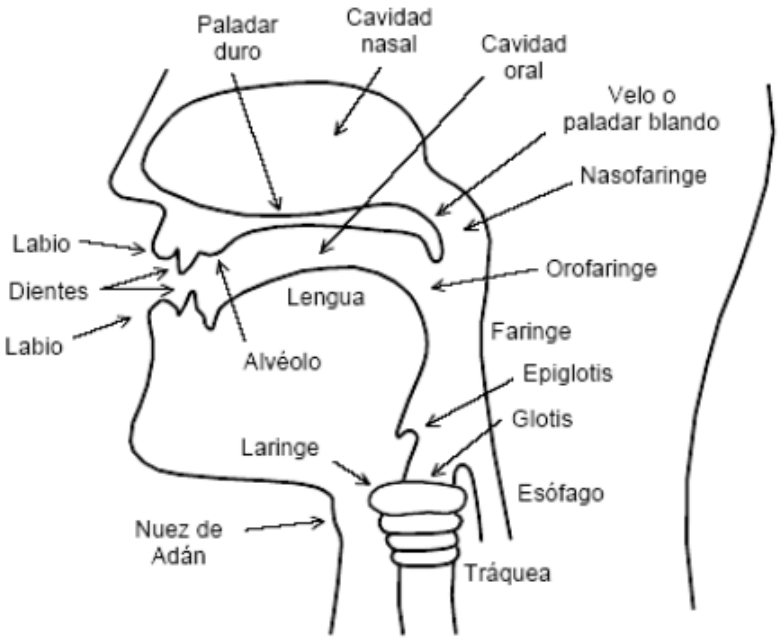


Figura 3.1: Corte esquemático del aparato fonatorio humano [12]

mecanismos principales: el filtrado y la articulación.

El filtrado actúa modificando el espectro del sonido. Tiene lugar en la faringe, la cavidad nasal, la cavidad oral y la cavidad labial. Las mismas constituyen resonadores acústicos que enfatizan determinadas bandas frecuenciales del espectro generado por las cuerdas vocales, conduciendo al concepto de formantes, es decir una serie de picos de intensidad en el espectro de la voz, ubicados en frecuencias o bandas de frecuencia que son bastante específicas para cada tipo de sonido.

La articulación es una modificación principalmente a nivel temporal de los sonidos, y está directamente relacionada con la emisión de los mismos y con los fenómenos transitorios que los acompañan. Está caracterizada por el lugar del tracto vocal en que ocurre, por los elementos que intervienen y por el modo en que se produce.

Resumiendo, el aire hace vibrar las cuerdas vocales, generando ondas periódicas (tonos) o ruido, dependiendo de la posición de la glotis. Las ondas siguen su recorrido a través de la faringe hacia la cavidad oral y/o nasal, hasta salir por los labios. En ese pasaje el espectro de las ondas cambia de forma, debido a las variantes en el diámetro de la laringe, la posición de la lengua, la mandíbula, los dientes y los labios. Este proceso corresponde a la articulación de los sonidos [20].

Una vez diferenciados los dos aspectos de excitación y articulación, se observa que la articulación está vinculada a los fonemas emitidos. Por otro lado, la excitación, y concretamente la fluctuación de la frecuencia fundamental y de la energía, aportan información sobre la expresión del lenguaje, la entonación, el sexo del locutor, el estado emocional, etc.

Ambos procesos se pueden considerar independientes el uno del otro. Por ejemplo, con una misma cadena de palabras y con diferentes formas de expresarlas se puede realizar una afirmación, una interrogación, etc.

Como se muestra en la Figura 3.2, la producción de la voz se puede modelar entonces como el filtrado de la excitación (señal periódica de las cuerdas vocales o señal de ruido producida por una constricción al paso del aire) por la transferencia del aparato fonador. Esta transferencia se considera un filtro lineal a efectos de simplificar el modelo; se representa en el dominio del tiempo como la convolución entre la señal de excitación y la respuesta impulsiva del aparato fonador, es decir:

$$y[n] = u[n] * h[n] \tag{3.1}$$

donde $u[n]$ corresponde a la excitación, $h[n]$ representa a la transferencia impuesta por el tracto vocal, y finalmente y[n] es la señal de audio resultante.

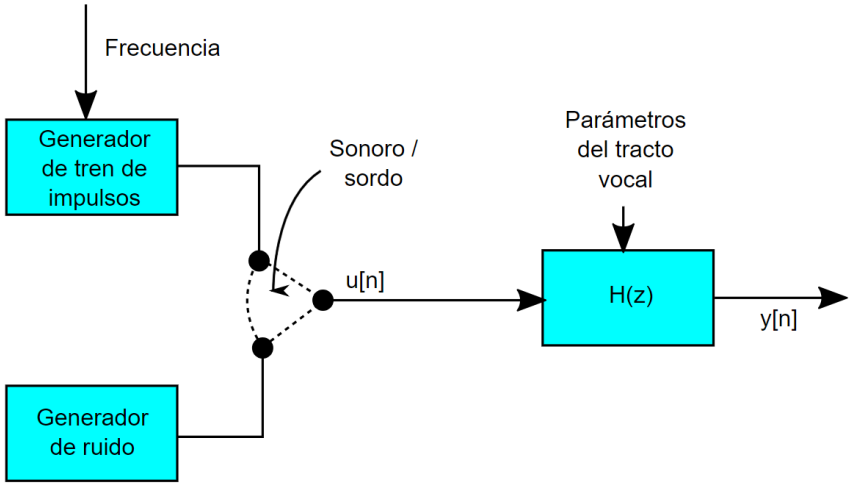


Figura 3.2: Modelo de producción de voz [17]

3.2. Coeficientes Cepstrales

Como se expuso en la ecuación 3.1, la señal de voz $y[n]$ se puede modelar como la convolución de la excitación glotal $u[n]$ con el tracto vocal visto como un filtro $h[n]$ lineal e invariante en el tiempo para tiempos cortos.

La señal h[n] posee información correspondiente a la articulación para cada fonema expresado, y por tanto es de gran interés obtener una descripción de h[n]. Sin embargo, la señal disponible es y[n], y h[n] aparece convolucionada con u[n]. Es decir, en principio h[n] y u[n] están 'mezcladas' y no es trivial la identificación de cada una de ellas.

Se puede realizar la separación tomando la Transformada de Fourier (TF) de la señal, luego calculando el logaritmo de la magnitud del espectro, y finalmente realizando la TF inversa.

$$y[n] = u[n] * h[n]$$ Al aplicar la TF,

$$Y(e^{j\theta}) = U(e^{j\theta})H(e^{j\theta})$$ la convolución pasa a ser un producto.

$$\log|Y(e^{j\theta})| = \log|U(e^{j\theta})| + \log|H(e^{j\theta})|$$ Luego se toma el logaritmo del módulo

$$c[n] = c_u[n] + c_h[n]$$ y se realiza la IFT.

La variable independiente de la función $c[n]$ obtenida es una variable temporal, ya que corresponde a la TF inversa del logaritmo del espectro, pero contiene información de frecuencia. Por este motivo se acuñó el término cepstrum, cambiando el orden de las letras de la palabra spectrum(espectro, en inglés). El dominio temporal del cepstrum se denomina quefrency. Dado que la función $c[n]$ es la adición de dos componentes, si estas son disjuntas en el dominio de las quefrencies, se pueden separar mediante un simple filtrado (llamado liftrado).

El cepstrum es útil ya que separa la fuente del filtro:

- Si nos interesa la excitación glotal, debemos quedarnos con los coeficientes más altos. Se pueden utilizar, por ejemplo, para estimar las frecuencias formantes.
- Si nos interesa una descripción del tracto vocal, nos quedamos con los coeficientes más bajos. Se utilizan para el reconocimiento de voz.

Estos coeficientes son llamados coeficientes cepstrales, y analíticamente se obtienen de la sigiente manera:

$$c(\tau) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log(|Y(e^{j\omega})|)e^{j\omega\tau}d\omega \tag{3.2}$$

3.3. MFCCs

Los MFCCs (en inglés Mel-Frequency Cepstral Coefficients) son una representación de una señal de audio basada en la percepción auditiva humana que utiliza la información espectral y es muy popular en el área de la detección y reconocimiento del habla.

3.3.1. Escala Mel

La Escala Mel, propuesta por Stevens, Volkman y Newmann en 1937 [19], es una escala musical perceptual de tonos juzgados como intervalos equiespaciados

por parte de observadores. El punto de referencia entre esta escala y la frecuencia normal se define equiparando un tono de 1000 Hz, 40 dBs por encima del umbral de audición del oyente, con un tono de 1000 mels. Por encima de 500 Hz, los intervalos de frecuencia espaciados exponencialmente son percibidos como si estuvieran espaciados linealmente. La fórmula para pasar de f en Hz a m en $mels$ es:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Los MFCC son una variante de la ecuación 3.2. En primer lugar, el logaritmo no se aplica directamente al espectro, sino que previamente se multiplica la DFT por un banco de filtros triangulares espaciados según la escala mel. Luego se calcula la energía asociada a cada filtro, y se aplica la función logaritmo. La segunda modificación consiste en la utilización de la DCT en lugar de la DFT para calcular la IFT. Esto busca lograr una reducción en el número de coeficientes del cepstrum debido a su propiedad de "compactación de energía", resaltando los de más bajas quefrencias. En resumen, los MFCC se calculan con los siguientes pasos:

1. Cálculo de la DFT de la señal enventanada
2. Multiplicación por un banco de filtros triangulares espaciados según la escala Mel.
3. Cálculo de la energía en cada filtro
4. Cálculo de la DCT del logaritmo de la energía en cada filtro

$$E[l] = \sum_{k=0}^{K-1} \left[Y[k]V_l[k]\right]^2, \quad l = 0, ..., L-1$$

Donde V_l representa uno de los L filtros y $E[l]$ es la energía en el filtro l.

Finalmente se obtienen los coeficientes cepstrales en las frecuencias de Mel:

$$\text{MFCC}[i] = \sum_{k=0}^{K-1} \log(E[k]) \cos\left(\frac{2\pi}{K} ki\right), \quad i = 0, ..., K-1$$

3.3.2. Implementación del cálculo de los MFCCs

Al realizar la extracción de características, cada fragmento de audio será representado por una matriz que a cada ventana de tiempo le asigna un vector de características MFCC.

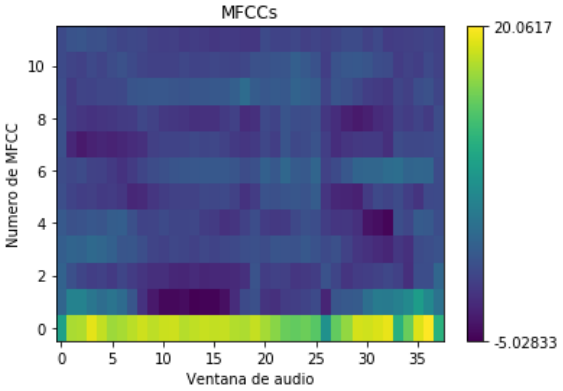
Para realizar el cálculo de los MFCCs de las ventanas de audio se utilizó una implementación encontrada en Essentia [3]. No hay definida una implementación estándar para el cálculo de MFCCs. La función que provee Essentia permite modificar los parámetros clave(número de Bandas Mel, número de coeficientes Mel,etc).

Los parámetros seleccionados para nuestra implementación son:

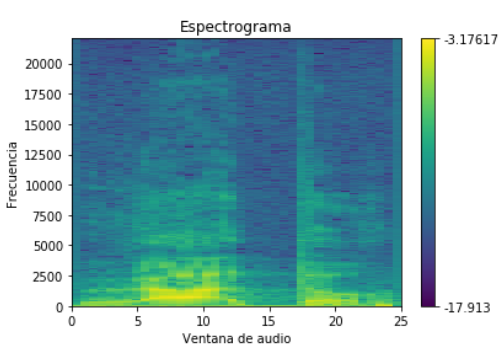
- numberBands=26, Número de bandas Mel en el filtro
- numberCoefficients=13, Número de coeficientes cepstrales
- dctType=2, Algoritmo DCT-II (ver Apéndice E)
- logType='log', Método de compresión en las bandas Mel

El tamaño de ventana que elegimos utilizar es de $30ms$, un número de Bandas Mel igual a 26, y 13 coeficientes MFCCs. El primer coeficiente cepstral, que corresponde a la potencia de la señal(no aporta información para el reconocimiento de voz), se desecha, dando lugar a 12 coeficientes. No obstante, más adelante se evaluarán distintos valores del número de MFCCs, como parámetro clave que afecta el desempeño del sistema.

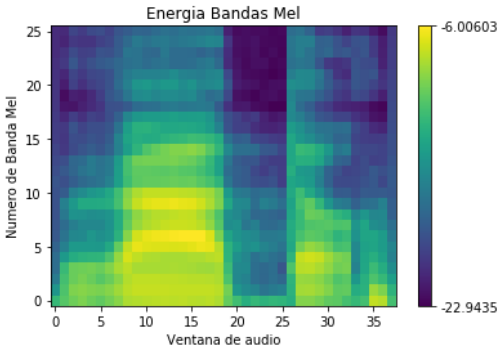
Essentia [3] dispone de una librería para Python dedicada al procesamiento de audio. La librería también cuenta con herramientas para el cálculo del Espectrograma y la Energía en las Bandas Mel.



(a) MFCCs



(b) Espectrograma



(c) Energía en las Bandas Mel

Figura 3.3: Estudio de los MFCCs para la palabra "gato"

En la Figura 3.3, se pueden apreciar todas estas magnitudes para una grabación de la palabra "gato". Se puede apreciar en todas las imágenes las diferencias en las magnitudes a lo largo de la palabra, y se pueden distinguir 4 regiones que correspondería a los cuatro fonemas que la forman.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Medida de similitud entre secuencias temporales

Para el sistema propuesto, es clave poder comparar dos fragmentos de audio y decidir si estos son dos instancias de la misma palabra, o si son palabras distintas. Con este propósito es necesario poder cuantificar qué tan parecida es una secuencia de otra. Se eligió utilizar una técnica muy empleada para comparar dos secuencias que dependen del tiempo, Dynamic Time Warping(DTW). Esta técnica, a su vez, requiere utilizar una medida de distancia espectral que compare la similitud entre dos fragmentos cortos de audio. Para esto último se consideraron varias técnicas y se optó por utilizar la distancia cepstral ponderada por seno.

4.1. Medidas de distancia espectral

Como se describió en el capítulo 3, las palabras son representadas como matrices formadas por vectores de coeficientes cepstrales MFCCs. Para comparar qué tan parecidos son dos vectores de características entre sí, se pueden emplear diferentes medidas de distancia.

Una medida de distancia d debe cumplir las siguientes tres propiedades matemáticas:

- No negatividad: $d(a,b) \geq 0$ y $d(a,b) = 0$ sii $a = b$
- Simetría: $d(a,b) = d(b,a)$
- Desigualdad triangular: $d(a,b) \leq d(a,c) + d(c,b)$

válido para todo a, b, c vectores de características.

A continuación se describen algunas medidas de distancia.

4.1.1. Distancia espectral logarítmica

La intensidad sonora percibida subjetivamente por el humano es aproximadamente logarítmica. Esto inspira el uso de distancias espectrales logarítmicas, ya

que se busca obtener una relación cercana a la percepción de la diferencia entre sonidos.

Si consideramos dos fragmentos de audio, representados por sus respectivos espectros de potencia (magnitud de TF al cuadrado) $S(f) = X^2(f)$ y $S'(f) = X'^2(f)$, la distancia entre estos dos espectros se puede definir como:

$$V(f) = log(S(f) - log(S'(f))$$

Esta medida es llamada Diferencia Espectral Logarítmica. Tomando esta medida como base, se puede definir un conjunto de normas L_p al definir:

$$d(S, S')^p = \int_{-\infty}^{\infty} |V(f)|^p df \tag{4.1}$$

En particular, una variación que ha encontrado aplicación en muchos sistemas de procesamiento de voz es la llamada Distancia Logarítmica Espectral RMS [17], la cual se define al tomar $p = 2$.

4.1.2. Distancia cepstral

El espectro de potencia $S(\omega)$ de una señal se puede expresar de la siguiente manera

$$\log(S(f)) = \sum_{n=-\infty}^{\infty} c_n e^{-jn2\pi f},$$

donde $c_n = c_{-n}$ son reales y son los coeficientes cepstrales vistos previamente en el capítulo 3.

Para dos espectros $S(f)$ y $S'(f)$, aplicando el teorema de Parseval, podemos relacionar la distancia cepstral con la distancia logarítmica espectral rms:

$$d_2^2(S, S') = \int_{-\infty}^{\infty} |\log(S(f)) - \log(S'(f))|^2 df = \sum_{n=-\infty}^{\infty} (c_n - c'_n)^2, \tag{4.2}$$

donde c_n y c'_n son los coeficientes cepstrales de $S(f)$ y $S'(f)$ respectivamente.

Esta técnica, por tanto, consiste en una aproximación a la distancia presentada anteriormente que reduce sustancialmente los tiempos de cómputo, ya que el número de coeficientes cepstrales que se utilizan es mucho menor al número de coeficientes en una representación espectral clásica (entre 10-15 para el primero, y cientos en el último).

4.1.3. Distancia cepstral ponderada

Se puede demostrar [8] que, considerando a las señales de voz como procesos estacionarios(en tiempo corto), los coeficientes cepstrales, excepto c_0, tienen media

nula y varianza inversamente proporcional al cuadrado del índice del coeficiente, es decir $E\{c_n^2\} \approx \frac{1}{n^2}$.

Si se incorpora el factor n^2 en la distancia cepstral para normalizar la contribución de cada coeficiente cepstral, la distancia expresada en la ecuación 4.2 quedará como:

$$d_{2W}^2(S, S') = \sum_{n=-\infty}^{\infty} n^2(c_n - c'_n)^2 = \sum_{n=-\infty}^{\infty} (nc_n - nc'_n)^2 \tag{4.3}$$

Como se verá más adelante, esta forma de cálculo tiene, para nuestro sistema, un desempeño mejor respecto a la medida anterior.

4.1.4. Distancia cepstral ponderada por seno elevado

El procedimiento de ponderar la distancia cepstral por medio de una función $w(n)$ se puede generalizar escribiendo la distancia cepstral de la ecuación 4.3 como:

$$d_{2W}^2(S, S') = \sum_{n=-\infty}^{\infty} (w(n)c_n - w(n)c'_n)^2 \tag{4.4}$$

$$d_{cW}^2(x, y) = \sum_{n=-\infty}^{\infty} (w(n)c_n^x - w(n)c_n^y)^2 \tag{4.5}$$

$$d_c^2(x, y) = \sum_{n=-\infty}^{\infty} (c_n^x - c_n^y)^2$$

Se puede demostrar [17] que los coeficientes cepstrales de mayor índice presentan mayor variabilidad. Esto sugiere que suprimir los coeficientes cepstrales de mayor orden debería dar una medida más confiable al comparar la diferencia espectral. A su vez, los coeficientes cepstrales bajos presentan variabilidad causada por variaciones en la transmisión, características del hablante, esfuerzos vocales y otros factores.

Por tanto, se puede diseñar una función $w(n)$ buscando reducir estas variabilidades que no aportan información en cuanto al reconocimiento de las palabras, para discriminar sonidos de forma más efectiva. Una opción, es considerar $w(n)$ en la forma de seno elevado:

$$w(n) = \begin{cases} 1 + h\sin(\frac{n\pi}{L}) & \text{para } n = 1, 2, ..., L \\ 0 & \text{para } n \leq 0, n > L \end{cases} \tag{4.6}$$

, donde L es el largo del vector de coeficientes cepstrales, y h es usualmente elegido $L/2$.

4.1.5. Elección de medida de distancia

En las secciones anteriores se vio cuatro medidas de distancia diferentes, expuestas en cierto orden. Se puede apreciar que el grado de complejidad en el análisis que propone cada una de estas soluciones va en aumento desde la primera a la última, buscando en cada paso lograr una mejora respecto a la propuesta anterior. Con esta idea, se considera que la medida de distancia cepstral ponderada por el seno elevado es, entre las propuestas, la herramienta más sofisticada para discriminar vectores de características de voz. Buscando corroborar la elección recurrimos al análisis visual de la matriz de distancias.

La matriz de distancias se confecciona calculando la distancia entre cada vector de características de una fragmento de audio con cada vector de características del otro fragmento de audio con el que se quiere comparar. Para la representación visual de esta matriz, para cada distancia se asigna un color que la representa.

En el caso particular de calcular la matriz de distancias de una señal de audio consigo misma, la llamamos **matriz de autosimilitud**.

En la Figura 4.1 se puede ver cuatro marices de autosimilitud correspondientes a una misma grabación de la secuencia de palabras "gato-jugo-gato". Estas matrices se diferencian solamente en la elección de la distancia espectral empleada.

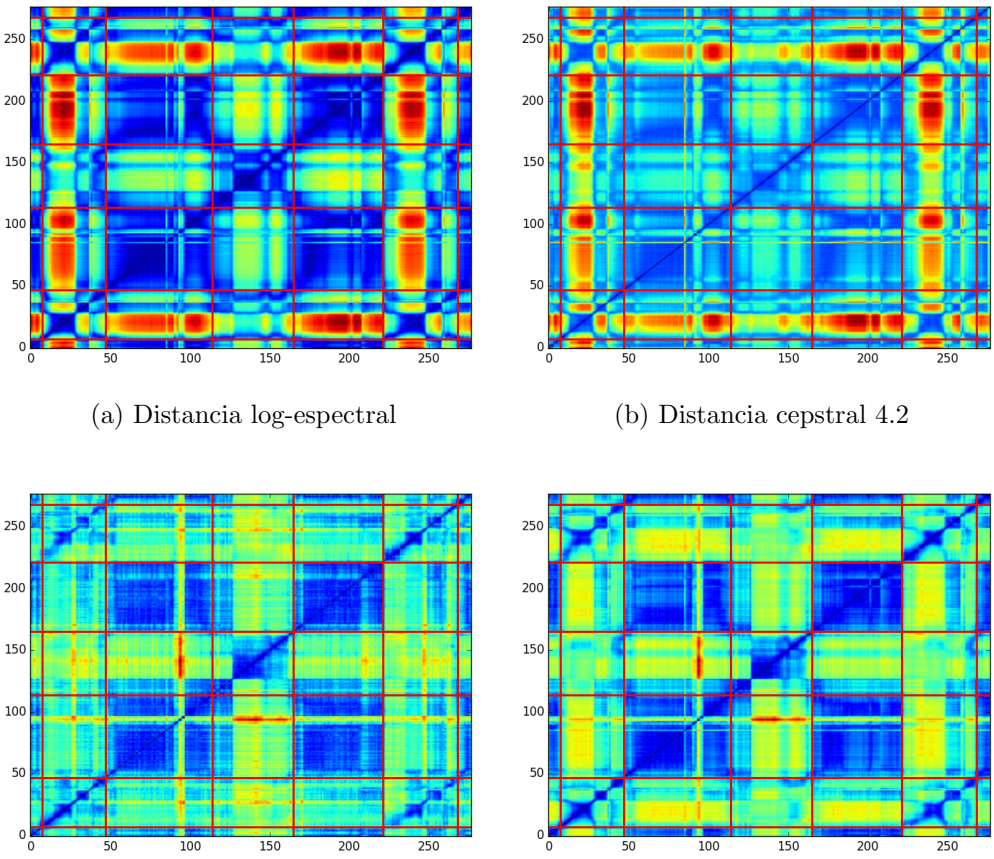
En una representación gráfica de una matriz de similitud se deben representar valores de tres dimensiones: trama de audio de señal 1, trama de audio de señal 2, y distancia entre dichas tramas. Para representar la intensidad, se utiliza una convención de colores similar a un mapa de calor. Los colores fríos representan distancias pequeñas, y al acercarse a los colores cálidos la distancia aumenta. Es decir, en un extremo el color azul (frío) indica distancia nula, mientras que el color rojo (cálido) indica distancias grandes. Como se puede ver, todas las matrices muestran en su diagonal principal una marcada linea azul oscuro. Esta línea corresponde a la distancia entre tramas correspondientes a un instante de tiempo t consigo mismo, y dado que $d(t, t) = 0$ (como se vio en las ecuaciones 4.1), esta diagonal indica distancia cero.

De un modo similar, cuando se pueden ver diagonales secundarias marcadas en color azul, estas indican que las tramas correspondientes son muy similares, pudiendo significar la repetición de una palabra. Esto se puede ver claramente en la Figura 4.1 para las posiciones donde se comparan la primer y tercera palabras, mientras que se aprecia que al comparar la segunda palabra con cualquiera de las otras, no se ve una diagonal, como es de esperarse al comparar palabras diferentes.

Más allá de estas observaciones, la inspección visual no permite tomar una buena decisión sobre qué medida de distancia es más adecuada. Por lo que en esta etapa elegimos proseguir bajo el supuesto de que la distancia cepstral ponderada por seno elevado es la más apropiada.

4.1.6. Implementación en C

Para la primer implementación de la función de distancia entre tramas de audio, se utilizó el lenguaje Python, como para la realización de todo el sistema.



(a) Distancia log-espectral

(b) Distancia cepstral 4.2

(c) Distancia ponderada por n 4.3

(d) Distancia ponderada por seno 4.6

Figura 4.1: Matrices de autosimilitud para la secuencia "Gato-Jugo-Gato"

Los valores en los ejes corresponden al número de trama. Las líneas rojas indican el inicio y el final de las palabras.

Como se verá en la sección 4.4, la distancia entre tramas se invoca miles de veces por cada corrida del sistema, y se vio que la mayor parte del tiempo de procesamiento era ocupado por esta función. En un principio el tiempo de procesamiento no parecía algo importante ya que no hay ningún requerimiento en este sentido. Empíricamente se vio que la realización de evaluación de una grabación de audio, podía tomar entre 1 y 4 minutos (utilizando computadoras domésticas de 4 núcleos).

En la etapa final del proyecto se hace necesario entrenar y validar el sistema (ver capítulo 6). Para esto es necesario que toda la implementación del sistema corra un gran número de veces, realizando modificaciones en el valor de algunos parámetros y comparando los resultados obtenidos. Realizar esto con la implementación de distancia hecha en Python suponía semanas de procesamiento. El lenguaje de programación C es varias veces más rápido que Python para realizar operaciones

entre vectores y matrices. Al implementar este módulo en C se logró reducir el tiempo de procesamiento del sistema entre 30 y 50 veces. Esto nos permitió realizar el entrenamiento y validación del sistema en aproximadamente 4 días (utilizando dos computadoras).

4.2. Distancia de Edición

En algunas aplicaciones es de utilidad considerar el problema de cómo cambiar una secuencia de símbolos en otra, con el menor número de ediciones. Por ejemplo, esto tiene aplicación en los editores de texto que identifican una palabra con errores de escritura y sugieren una o varias palabras para corregirlo. Las palabras que se sugieren se identifican como aquellas que tienen el menor costo o distancia de edición. Las operaciones de edición permitidas son insertar, eliminar y cambiar (un símbolo por otro símbolo) y hay un costo asociado con la realización de cada edición. El problema de edición es identificar una secuencia de operaciones de edición que transforme una palabra en otra con un costo mínimo.

Por ejemplo, definidos los costos de insertar una letra (ci), eliminar una letra (cd), y cambiar una letra (cr), $ci = cd = cr = 1$, el costo óptimo de transformar *intención* en *ejecución* es 5:

I	N	T	E	*	N	C	I	Ó	N
|	|	|	|	|	|	|	|	|	|
*	E	J	E	C	U	C	I	Ó	N

1. Se elimina la letra “I”.
2. Se cambia la letra “N”por la “E”
3. Se cambia la letra “T”por la “J”
4. Se inserta la letra “C”
5. Se cambia la letra “N por la “U”

Como parte de este proyecto se implementa una solución a este problema utilizando programación dinámica. Inicialmente esto se realizó como un primer acercamiento al problema de comparar fragmentos de audio, ya que la distancia de edición se puede considerar como una simplificación de ese problema. Además, la función implementada se utilizará para determinar una medida de desempeño del sistema.

4.3. Programación dinámica

La programación dinámica es un método de diseño de algoritmos que puede ser utilizado cuando la solución de un problema puede ser vista como el resultado de una secuencia de decisiones.

Para algunos problemas que pueden ser vistos de esta forma, se puede tomar las decisiones de una en una, nunca tomando una decisión equivocada, y así obtener una secuencia de decisiones óptima, es decir una secuencia de decisiones con costo mínimo. Para otros problemas no es posible determinar una secuencia de decisiones óptima de esta manera. Una forma de enfrentar este tipo de problemas

sería calcular todas las secuencias de decisiones posibles y luego elegir la mejor, pero esto puede ser prohibitivo por los requerimientos de tiempo y recursos.

La programación dinámica reduce el número de cálculos al evitar calcular secuencias de decisiones que no es posible que sean óptimas. Se obtiene una secuencia de decisiones óptimas utilizando convenientemente el principio de optimalidad. En Apéndice B se explica este principio.

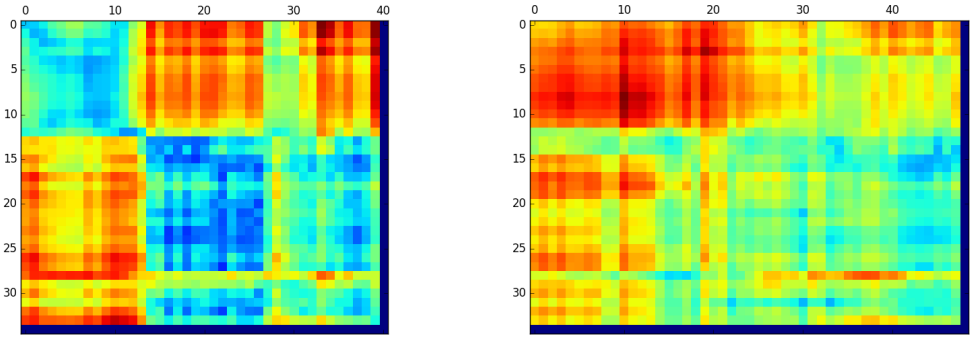
4.4. Dynamic Time Warping

El Alineamiento Temporal Dinámico, o DTW (Dynamic Time Warping), es una técnica utilizada para encontrar la alineación óptima entre dos secuencias dadas. Este método realiza una deformación no lineal de las secuencias, buscando alinear una a la otra.

Originalmente, DTW se utilizó para comparar patrones de voz para reconocimiento automático del habla; como en el caso de este trabajo. En campos como minería de datos y búsqueda y recuperación e información, DTW ha sido aplicada exitosamente para manejar automáticamente deformaciones temporales y diferentes velocidades asociados a datos que dependen del tiempo [14].

El objetivo de DTW es comparar dos secuencias dependientes del tiempo $X = x_1, x_2, ..., x_N$ de longitud $N \in \mathbb{N}$ e $Y = y_1, y_2, ..., y_N$ de longitud $M \in \mathbb{N}$. Estas secuencias pueden ser señales discretas o, en general, secuencias de vectores de características muestreadas en puntos equidistantes en el tiempo. De aquí en adelante, se trabajará con el espacio de características $\mathbb{F}$, donde $x_n, y_n \in \mathbb{F}$ para $n \in [1 : N]$ y $m \in [1 : M]$. Para comparar dos características diferentes $x, y \in \mathbb{F}$, se necesita una medida de distancia, la cual se define como una función: $d : \mathbb{F} \times \mathbb{F} \to \mathbb{R}_{\geq 0}$. Típicamente $d(x, y)$ es pequeña cuando x e y son similares entre sí, y $c(x, y)$ es grande en caso contrario. Evaluando la medida de distancia para cada par de elementos de X e Y se obtiene una matriz de distancias $D \in \mathbb{R}^{N \times M}$ definida por $D(n, m) = d(x_n, y_m)$. Luego se busca una alineación entre X e Y que tenga distancia total mínima. Intuitivamente, esta alineación óptima, correrá a lo largo de un valle de costo bajo en la matriz D.

En la Figura 4.2 se pueden ver algunas matrices de distancias. En (a) se puede ver como quedan delimitadas las vocales en "*gato*", las cuales ocupan la mayor cantidad de tiempo. La distancia entre tramas para estos bloques bien marcados está indicada con tonos de azul, es decir, distancias pequeñas. En (b) se compara una instancia correspondiente a la palabra "*gato*" con una que corresponde a "*jugo*". En este caso ya no se distinguen estructuras comunes y los colores son más cálidos, indicando distancias mayores entre vectores de características. En particular la esquina superior izquierda muestra un bloque de color rojo que corresponde a comparar el segmento de señal que corresponde a la sílaba "*ga*" con el segmento correspondiente a la sílaba "*ju*". Estas sílabas tienen representación muy distinta en el espacio de características, por tanto la distancia entre tramas es mayor (color rojo).



(a) Matriz de distancias de diferentes instancias de "*gato*"

(b) Matriz de distancias entre instancias de "*gato*" y "*jugo*"

Figura 4.2: Ejemplos de matrices de distancias

Un camino de alineamiento (warping path) es una secuencia $p = p_1, p_2, ..., p_L$ con $p_l = (n_l, m_l) \in [1:N] \times [1:M]$ para $l \in [1:L]$ que satisface las siguientes tres condiciones:

1. Condición de borde: $p_1 = (1,1)$; $p_L = (N, M)$
2. Condición de monotonía: $n_1 \leq n_2 \leq ... \leq n_L$ y $m_1 \leq m_2 \leq ... \leq m_L$
3. Condición de tamaño de paso: $p_{l+1} - p_l \in \{(1,0), (0,1), (1,1)\}$ para $l \in [1 : L-1]$

Un camino de alineamiento define una relación entre dos secuencias X e Y asignando a cada elemento de X un elemento de Y. La condición de borde fuerza a que el primer elemento de X esté alineado con el primero de Y, así como el último elemento de X con el último de Y, es decir que la alineación aplica a las secuencias enteras. La condición de monotonicidad refleja el requerimiento de ser coherentes con la dependencia temporal. Si un elemento en X precede a un segundo, esto debe corresponderse con los elementos de Y y viceversa. Finalmente, la condición de tamaño de paso expresa un tipo de continuidad, forzando a que ningún elemento de X o Y sea omitido, y que no haya repeticiones en el alineamiento (en el sentido de que ningún par contenido el camino de alineamiento será repetido).

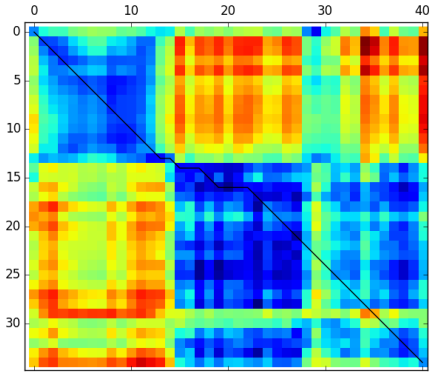
La distancia total $d_p(X, Y)$ de un camino de alineamiento p entre X e Y con respecto a la medida de distancia d se define como

$$d_p(X, Y) = \sum_{l=1}^{L} d(x_{n_l}, y_{m_l})$$

Por lo tanto un camino de alineamiento óptimo entre X e Y es un camino de alineamiento $p*$ cuya distancia total es el mínimo dentro de todos los posibles

caminos de alineamiento. La distancia DTW entre X e Y, $DTW(X,Y)$, se define como la distancia total de $p*$:

$$DTW(X,Y) = c_{p*}(X,Y) = min\{c_p(X,Y) \mid p \text{ es un camino de alineamiento}\}$$



(a) Matriz de distancias y camino de alineamiento entre diferentes instancias de "*gato*"

Figura 4.3: Ejemplo de camino de alineamiento óptimo entre dos instancias de "*gato*"

En la Figura 4.3 se puede ver un ejemplo de camino de alineamiento al comparar dos instancias en las que el hablante produce la misma palabra. Se puede ver como el camino de alineamiento busca, dentro de lo posible, atravesar las zonas de menor distancia entre tramas de audio.

Para determinar un camino óptimo $p*$, se podrían calcular todos los caminos de alineamiento posibles entre X e Y. Este procedimiento, sería muy complejo computacionalmente.

Normalmente este problema se resuelve con un algoritmo basado en programación dinámica. Con este cometido se definen las secuencias $X(1:n) = x_1, ..., x_n$ para $n \in [1:N]$ e $Y(1:m) = (y_1, ..., y_m)$ para $m \in [1:M]$. Además se define $D(n,m) = DTW(X(1:n), Y(1:m))$.

Los valores $D(n,m)$ definen una matriz $n \times m$, conocida como matriz de costo acumulado o matriz de distancia acumulada(D). La última entrada de esta matriz es $D(N,M) = DTW(X,Y)$. El teorema 1 muestra cómo se puede computar D eficientemente.

Es importante observar que el valor de $D(N,M)$ será mayor cuando se comparan palabras largas, en contraste a la situación donde se comparan palabras cortas. Esto es porque $D(N,M)$ se computa como la suma de las distancias en el

camino de alineamiento óptimo, y para palabras de mayor largo, esta suma estará compuesta por un mayor número de factores.

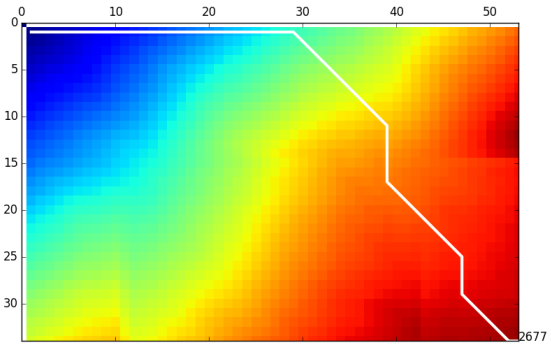
En la Figura 4.4 se muestran tres ejemplos de matrices de distancia acumulada. En (a) se puede ver que al comparar instancias de palabras claramente diferentes, la distancia acumulada crece de tal modo que no existen valles que encaucen al camino de alineamiento óptimo. La distancia resultante es relativamente grande considerando que las palabras comparadas no son particularmente largas. En (b) y (c) existen valles de menor distancia acumulada, y se puede ver como los caminos de alineamiento óptimos siguen esos valles. Si se compara la distancia acumulada en (a) con la distancia acumulada en (b), donde se comparan palabras de largos similares, se puede ver en (a) una distancia total menor significativamente menor. En contraste, al comparar (a) con (c), las distancias acumuladas son casi iguales. Esto pone en evidencia la necesidad de tener en cuenta el largo de las palabras a comparar para normalizar una medida de distancia entre palabras. Se volverá sobre este último aspecto en la sección 4.5.

Teorema 1. *La matriz de costo acumulado D satisface las siguientes identidades:* $D(n,1) = \sum_{k=1}^{n} c(x_k, y_1)$ *para* $n \in 1:N$*;* $D(1,m) = \sum_{k=1}^{m} c(x_1, y_k)$ *para* $m \in 1:M$*, y*

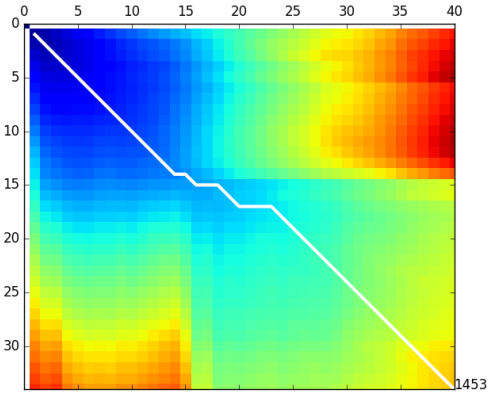
$$D(n,m) = min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$$

para $1 < n \leq N$ *y* $1 < m \leq M$*. En particular,* $D(N,M)$ *puede ser computado con* $O(NM)$ *operaciones.*

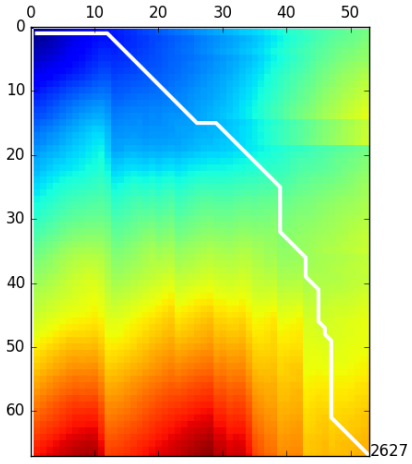
Este teorema permite un calculo recursivo de la matriz D. Y una vez hallada esta matriz, la distancia entre las dos secuencias $DTW(X,Y)$ queda determinada por la última entrada de la matriz D, $D(N,M)$.



(a) Matriz de distancia acumulada entre instancias de "*gato*" y "*queso*"



(b) Matriz para dos instancias de "*gato*"



(c) Matriz de distancia acumulada entre "*queso*" y "*queso*"

Figura 4.4: Ejemplos de matrices de distancia acumulada y caminos de alineamiento óptimos. En la esquina inferior derecha de cada gráfica se puede ver la distancia acumulada final $D(N, M)$.

4.5. Implementación de la función de DTW

El algoritmo de DTW (Dynamic Time Warping) compara las tramas de una palabra p_M, de largo M tramas, con las de la palabra p_N de largo N tramas. Para comparar dos tramas, se utiliza una función de distancia, la cual es un parámetro de entrada en la función DTW. De este modo es posible confeccionar una matriz de distancias (d) de tamaño MxN. Cada elemento (m, n) de la matriz d es la distancia entre la ventana m perteneciente a la palabra p_M y la ventana n perteneciente a la palabra p_N.

A partir de esta matriz de distancias es que se construye una matriz de distancias acumuladas D que permite encontrar el camino más corto entre las palabras comparadas. Siguiendo las condiciones borde, de monotonía y de tamaño

de paso (explicadas en 4.4). La matriz de distancia acumulada D, de tamaño $(M+1) \times (N+1)$, se inicializa como:

$$D(i,j) = \begin{cases} \infty & i = 0, j \neq 0 \text{ y } i \neq 0, j = 0 \\ 0 & \text{para todo el resto} \end{cases}$$

Se agrega una fila y una columna extra para $i = 0$ y $j = 0$ respectivamente con distancia infinita. Esto permite que el algoritmo de la construcción de la matriz de distancias acumuladas no intente usar palabras de largo nulo, inexistentes, y al hacer infinita la distancia nos aseguramos que estos elementos no formen parte del cálculo de la distancia acumulada en la construcción de la matriz D. No obstante, se toma $D(0,0) = 0$ así $D(1,1)$, que es el primer valor calculado, puede calcularse a partir de este y al ser nulo no agrega distancia adicional. Esto también sirve para cumplir la condición de borde.

La distancia acumulada para la posición (i,j) lo calculamos mediante la siguiente ecuación:

$$D(i,j) = min\big(c_i d(i,j) + D(i-1,j),\ c_j d(i,j) + D(i,j-1) \\ c_r d(i,j) + D(i-1,j-1)\big)$$

Donde c_i, c_j y c_r son factores de peso locales asociados a avanzar en las direcciones horizontal, vertical y diagonal, respectivamente. También se aprecia como se cumple la condición de tamaño de paso, ya que cada elemento de D se calcula utilizando sus vecinos adyacentes.

Al final, el último elemento $D(M,N)$ corresponde al costo del camino de alineamiento óptimo.

Al comparar palabras largas, el camino más corto resulta de mayor longitud que en palabras cortas, entonces $D(M,N)$ es la suma de un mayor número de factores en palabras largas y por tanto $D(M,N)$ va a ser mayor para palabras largas. Para mitigar este efecto utilizamos el factor de normalización $(M+N+2)$

Finalmente, la distancia entre las palabras (DTW) se define como:

$$DTW(p_M, p_N) = \frac{D(M,N)}{M+N+2}$$

Notar que para $(c_i, c_j, c_r) = (1,1,1)$(DTW clásica), se favorece el avance en el sentido de la diagonal, ya que un paso en sentido diagonal(costo de una entrada de la matriz de distancias) corresponde a un movimiento horizontal y otro vertical(costo de dos entradas). Para contrarrestar esta preferencia, a menudo se utilizan valores mayores a 1 para c_i [14].

Es importante observar que c_i debe ser igual a c_j, ya que al ser DTW una medida de distancia, debe cumplir $DTW(X,Y) = DTW(Y,X)$. A su vez, el valor de $c_i = c_j$ no es importante en si mismo, sino que lo que juega un papel determinante es la relación que tiene este factor con c_r. Es por eso que definimos el factor F_{DTW}

como

$$F_{DTW} = \frac{c_r}{c_i} = \frac{c_r}{c_j} \tag{4.7}$$

La implementación de la función que realiza DTW toma 4 entradas y devuelve 2 salidas:

ENTRADAS:

1. Vector de MFCCs para cada ventana en la palabra 1 (p_M).
2. Vector de MFCCs para cada ventana en la palabra 2 (p_N).
3. Factor de proporción entre el costo de avance en sentido diagonal y sentido horizontal/vertical, en la matriz de costos acumulados(F_{DTW}).
4. Función de distancia espectral usada para realizar la medida entre los *frames* de las palabras.

SALIDAS:

1. La distancia entre palabras, la cual es la medida buscada.
2. Matriz de costos acumulados a lo largo del camino construido en el algoritmo.

4.6. Optimización de parámetros en la función de DTW

La función que implementa el DTW, como ya se describió en 4.5, tiene tres parámetros, c_i, c_j y c_r, que influyen en la construcción del camino más corto en la matriz de distancia acumulada y, por ende, en el resultado final que definimos como distancia entre las palabras que la forman. Estos parámetros corresponden a las constantes que multiplican a las distancias entre tramas al avanzar en sentido horizontal, vertical y diagonal en la construcción del camino. Cuanto más grande sea una de las constantes, será más costoso avanzar en la dirección correspondiente. Si se aumenta el valor de la constante en una dirección, hay dos escenarios posibles en cada paso del algoritmo:

1) El aporte $cd(i, j)$ a la distancia acumulada al avanzar en esa dirección sigue siendo menor al aporte de avanzar en otras direcciones, por lo tanto, el camino construido no cambia, no obstante como el costo aumentó, la distancia final obtenida en la DTW termina siendo mayor.

2) El aporte al avanzar en esa dirección ahora es mayor al aporte al avanzar en otras direcciones, entonces el camino construido cambia, por consiguiente, la distancia final obtenida en la función DTW termina siendo mayor.

Por lo tanto, en general, si se aumenta cualquiera de estas constantes, el DTW devuelve una distancia mayor. Como vimos en 4.5, estos tres factores se pueden reducir a un único factor $F_{DTW} = c_r/c_i = c_r/c_j$.

Para optimizar el valor de este parámetro tomamos en cuenta que para palabras iguales se busca una distancia chica y para palabras distintas se busca una distancia grande. Por lo tanto, se tomaron dos funciones que buscan cuantificar esta idea:

$$F_1 = \sum_{n=0}^{N} \sum_{m=0}^{N} \frac{DTW(m,n)(-1)^L}{K_i(L)+K_d(1-L)}$$

$$F_2 = \frac{\sum_{n=0}^{N} \sum_{m=0}^{N} DTW(m,n)(1-L)}{\sum_{n=0}^{N} \sum_{m=0}^{N} DTW(m,n)(L)}$$

K_i = número de iteraciones para palabras iguales
K_d = número de iteraciones para palabras distintas

$$L(m,n) = \begin{cases} 1 & \text{si } m,n \text{ palabras iguales} \\ 0 & \text{si } m,n \text{ palabras distintas} \end{cases}$$

Las constantes K_i y K_d en F_1 funcionan de normalización y L es un parámetro que indica si se está midiendo distancias para palabras iguales o distintas.

La primera función (F_1) suma la DTW de las palabras distintas y resta la DTW de las palabras iguales. F_2 hace el cociente entre la suma de la DTW de todas las palabras distintas y la suma de la DTW de todas las palabras iguales. Como se puede ver, ambas funciones crecen cuánto más grande sea la distancia entre palabras distintas y también crecen cuánto más chica sea la distancia entre palabras iguales. El máximo de estas funciones indicaría el punto óptimo de operación para el DTW.

Para la optimización se utilizaron grabaciones de prueba, conteniendo varias instancias de cinco palabras distintas (aproximadamente 50 de cada una) 1.4, etiquetadas para poder comparar instancias de palabras distintas y de palabras iguales. Considerando la duración del test y la exactitud buscada en esta instancia, se decidió correrlo para los valores de c_r= [1.0, 1.2, 1.4, 1.6, 1.8, 2.0]. El mínimo valor para el test (1.0) se eligió considerando que, en general, se suele utilizar $c_r \geq 1$ y para valores mayores al máximo (2.0) el avance en diagonal pasa a ser muy costoso, por lo tanto la función DTW pierde versatilidad, ya que se impone, en cierta medida, que el avance sea siempre en horizontal o vertical. Los resultados obtenidos para las funciones F_1 y F_2 dados por el test confirman estas afirmaciones y se pueden apreciar en la tabla 4.1.

Como se aprecia, el máximo para la F_1 corresponde a c_r = 1,6 y el máximo para la F_2 corresponde corresponde a c_r = 1,4. Analizando más en fondo a las funciones propuestas, se concluyó que la que más debe aproximar a este óptimo buscado es la F_2.

c_r	1.0	1.2	1.4	1.6	1.8	2.0
F_1	11.43	13.45	14.85	15.23	14.92	14.47
F_2	41.31	42.13	42.18	41.23	39.90	38.86

Tabla 4.1: Test de Optimización para DTW

Esto se debe a que el aumento y disminución de c_r, tiende a respectivamente, aumentar y disminuir la distancia entre palabras. Como se planteó que las variaciones de las distancias para palabras distintas y para palabras iguales, que sean proporcionales entre sí, no influyan en la toma de decisión del óptimo para c_r, se considera que la F_2 es la mejor opción para encarar el problema, ya que trabaja con el cociente entre las distancias de ambas clases (iguales, distintas).

Por ejemplo, si se aumenta c_r obteniendo distancias mayores para las dos clases con un incremento proporcional entre sí, F_1 terminaría devolviendo mayor valor debido a la naturaleza de la función, ya que terminan teniendo más peso en la toma de decisión las distancias entre palabras distintas. En conclusión, el óptimo encontrado corresponde a $c_r = 1{,}4$ y para un test de optimización posterior, que abarque al sistema de punta a punta, se trabajará en un rango alrededor de este valor.

Capítulo 5

Técnicas de agrupamiento

En la solución del sistema se planteó un bloque dedicado al agrupamiento de palabras. El test RAN está compuesto siempre de seis palabras que se repiten cinco veces en orden aleatorio a lo largo de la secuencia. Este bloque busca dividir las cinco instancias de una misma palabra en un solo grupo, obteniendo al final seis grupos. Como forma de aclaración, lo que se agrupa en principio son los fragmentos de audio proporcionados por el segmentador, que pueden corresponder o no a palabras. No obstante, en varias situaciones nos referiremos a los fragmentos de audio como palabras sin considerar su diferencia.

El objetivo del agrupamiento es descubrir una estructura dentro de un conjunto de datos dividiéndolo en subconjuntos que muestren una cierta "coherencia". Con esto nos referimos a:

- "Coherencia": muestras dentro de un mismo grupo o cluster son más "parecidas" entre sí que a las muestras de otros clusters.
- Muestras "parecidas": noción de similitud o de distancia entre muestras.

Generalmente, los vectores de un mismo grupo (o *clusters*) comparten propiedades comunes. El conocimiento de los grupos puede permitir una descripción sintética de un conjunto de datos multidimensional complejo. De ahí su uso en minería de datos. Esta descripción sintética se consigue sustituyendo la descripción de todos los elementos de un grupo por la de un representante característico del mismo.

En algunos contextos, como el de la minería de datos, se lo considera una técnica de aprendizaje no supervisado puesto que busca encontrar relaciones entre variables descriptivas pero no la que guardan con respecto a una variable objetivo.

Existen dos grandes técnicas para el agrupamiento de casos:

- Agrupamiento jerárquico, que puede ser aglomerativo o divisivo (ej. *Agglomerative Clustering*).
- Agrupamiento no jerárquico, en los que el número de grupos se determina de antemano y las observaciones se van asignando a los grupos en función de su cercanía (ej. *K-Means*).

Se busca un algoritmo de agrupamiento no jerárquico, ya que se quiere aprovechar el hecho de que se conozca de antemano la cantidad de grupos y no se está buscando jerarquía en el agrupamiento de las palabras. Además solamente debe necesitar una matriz de distancias para llevar a cabo su función. Lo que se usa es una matriz de distancias entre los fragmentos de audio segmentados que es lo que se obtiene luego de aplicar DTW para compararlos. Spectral Clustering cumple con estos requisitos y es muy popular en el área de aprendizaje automático.

La implementación que se utiliza en el sistema se encuentra en la librería de *scikit-learn* para Python, la cual posee varias herramientas útiles para aprendizaje automático. A continuación se explica el algoritmo empleado en esta implementación y su justificación teórica.

5.1. Spectral Clustering

Spectral Clustering utiliza los valores propios de la matriz de afinidad de los datos para reducir la dimensionalidad de los vectores propios y realizar el agrupamiento en una dimensión menor. La matriz de afinidad W consiste en una evaluación cuantitativa de una similitud relativa de cada par de instancias que componen los datos. W_{ij} es la afinidad entre las instancias con índices i y j respectivamente. W cumple la propiedad de simetría ya que $W_{ij} = W_{ji}$

La afinidad W puede construirse de varias formas siempre que cumpla los siguientes requisitos:

1. Debe indicar una medida de similitud entre las instancias.

2. Para todo par de instancias i y j se debe cumplir $W_{ij} \geq 0$.

3. $W(i, j) = 0$ representa que las instancias i y j no están conectadas.

Con respecto al tercer punto, en nuestro caso donde todos las instancias están conectadas nunca se va a dar que para algún (i, j) ocurra que $W(i, j) = 0$. Las instancias están todas conectadas ya que la distancia entre palabras siempre es un valor finito. Como forma de análisis, el tercer punto indica que cuánto más pequeño sea W_{ij}, más distintas serán las instancias i y j. Si se tiene una matriz de distancias d positivas donde para distancias grandes se tiene mayor disimilaridad que para distancias pequeñas, contrario a lo que buscamos en la matriz de afinidad, un método popular de construir W es mediante un transformación Gaussiana de la forma:

$$W_{ij} = e^{\left(-\frac{1}{2\sigma^2} D_{ij}^2\right)},$$

donde D es la matriz de distancias entre fragmentos de audio σ controla qué tan rápido varía la afinidad W_{ij} con D_{ij}.

Spectral Clustering trata al problema como una partición de un grafo, sin tener consideración sobre las posibles formas de los grupos a encontrar. Sea el conjunto de datos:

$$X = \{x_1, x_2, ..., x_n\} \quad x_i \epsilon R^\alpha$$

Donde X son los puntos que forman el grafo y las uniones entre los mismos llevan un peso asociado definido por la matriz de afinidad W.

Un método para construir la partición es resolver el problema *MinCut*. Dado un número de grupos a encontrar k, el método consiste en realizar una partición $A_1, ..., A_k$ que minimice la siguiente función:

$$cut(A_1, ..., A_k) = \frac{1}{2} \sum_{i=1}^{k} S(A_i, \bar{A}_i)$$

Donde la función $S(A, B) = \sum_{i \epsilon A, j \epsilon B} W_{ij}$ es la suma de los pesos de las uniones entre vértices que conectan a las particiones A y B, y $\bar{A}_i$ es el complemento de A. Esto corresponde a encontrar la partición tal que los puntos en distintos grupos sean relativamente asimilares entre sí. El problema presentado en este enfoque es que tiende a crear grupos pequeños formados por puntos aislados en el grafo. Una forma de solucionar esto es normalizar la función propuesta:

$$Ncut(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{cut(A_1, ..., A_k)}{vol(A_i)}$$

Donde $vol(A) = \sum_{i \epsilon A} d_i$ es la suma del orden de los vértices en A y el orden de un vértice se define como $d_i = \sum_{n}^{j=1} W_{ij}$, por lo tanto $vol(A)$ mide el factor de normalización para A calculando la suma de los pesos de las uniones conectando a sus vértices. El criterio *Ncut* además de minimizar la similitud entre grupos, al igual que *MinCut*, también maximiza la similitud dentro de los grupos, ya que la similitud dentro los grupos se puede expresar como:

$$S(A, A) = vol(A) - cut(A, \bar{A}) \text{ [9]}$$

El problema de minimizar *Ncut* es de tipo *NP-hard*. Este tipo de problemas no se pueden resolver fácilmente. Por lo tanto para llegar al algoritmo de Spectral Clustering se tiene que relajar el problema, o sea asumir por lo menos una hipotésis adicional. Se puede demostrar [9] que encontrar una partición de un grafo de n vértices en k grupos minimizando *Ncut* es equivalente a hallar k vectores de la forma $h_j = (h_{1j}, ..., h_{nj})$ con $j = 1, ..., k$ y $h_{ij} = 1/vol(A_j)$ si el vértice v_i pertenece a A_j o es nulo de lo contrario. De este modo, los elementos de estos vectores indican a cuál grupo corresponde cada vértice del grafo. Para relajarlo se permite que los elementos de los vectores, en vez de solo tomar dos valores discretos, puedan

tomar un valor arbitrario en $\mathbb{R}$. En [9] se demuestra que para resolver este problema relajado primero se debe calcular los primeros k vectores propios (que corresponden a los valores propios más grandes) de la Matriz Laplaciana normalizada del grafo $L_{rw} = (D^{-1}W)$, donde D es una matriz diagonal $n \times n$ con el orden de los vértices $d_1, ..., d_n$ en la diagonal. Luego aplicar un algoritmo de agrupamiento (uno muy usado es *K-Means*) a los vectores fila de la matriz U de $n \times k$, cuyas columnas son los vectores propios de L_{rw} $v_1, ..., v_k$.

Este algoritmo es similiar al usado en [6], donde definen la Matriz Laplaciana como $L = D - W$ y calculan los vectores propios del problema generalizado $Lu = \lambda Du$, los cuales son iguales a los vectores propios de L_{rw} por lo tanto los algoritmos son análogos y producen el mismo agrupamiento.

En varias implementaciones, se encontrará que remplazan el Laplaciano usado L por $I - L$ donde I es la matriz identidad. Ya que los vectores propios no cambian y solo los valores propios asociados cambian (de λ_i a $1 - \lambda_i$), el problema no se ve afectado si se toman los últimos vectores propios del Laplaciano ya que en este caso se buscan los que tengan valores propios más chicos.

5.2. Detección de valores atípicos

Un valor atípico es una observación que es numéricamente distante del resto de los datos. Los valores atípicos pueden ser indicativos de datos que pertenecen a una población diferente del resto de las muestras establecidas, que en nuestro caso serían palabras no incluidas al test RAN. Como el número de grupos está definido previamente, sabemos que los valores atípicos se encuentran dentro de los grupos que idealmente corresponderían a una sola palabra.

Para considerar los casos donde la segmentación devuelve una palabra que no coincide con ninguna otra de las encontradas en el test RAN, se decidió eliminarlas ya que puede deberse a un error en la segmentación o un caso donde el hablante u otras fuentes externas articulan una palabra o producen un sonido fuera de lo acordado en el test.

Buscamos analizar cada grupo obtenido en el agrupamiento con el objetivo de averiguar si existe en el grupo algún elemento considerablemente distinto con los demás. Para lograr esto se calcula para cada elemento, o palabra, la distancia con cada uno del resto de los elementos en el grupo. La idea es eliminar los elementos cuyas distancias calculadas son bastante mayores a las distancias obtenidas para los demás elementos. Considerando que puede haber varios valores atípicos en un grupo, se decide considerar para cada elemento solamente los valores de distancia más pequeños obtenidos.

En esta parte hacemos las suposiciones, las cuales son:

1. En un grupo por lo menos hay 4 elementos que corresponden a la misma palabra.

2. No puede haber una cantidad mayor o igual a 3 de valores atípicos que sean parecidos entre sí.

Lo que nos permite hacer la primera suposición es el hecho de que los test RAN contienen 6 instancias de cada palabra por lo que tomar 4 encontradas como mínimo parece razonable. La segunda suposición viene del hecho de que tomar un mínimo menor a 4 puede generar complicaciones ya que existe la posibilidad de la presencia de muletillas en el habla o ruidos relativamente parecidos. Spectral Clustering puede probablemente terminar agrupando estos casos planteados en un mismo grupo junto a otros elementos que es esperable que correspondan a una misma palabra del test RAN. En el caso, por ejemplo, de que un hablante haya dicho la palabra *perdón* 4 veces termina ocasionando que no haya eliminación de estos elementos del grupo en el que fueron posicionados.

A partir de estas hipótesis planteadas, se decide comparar el segundo valor menor obtenido en las distancias para cada elemento con un umbral elegido experimentalmente. Si resulta mayor al umbral, al elemento se lo considera un valor atípico y se elimina del grupo, y en el caso contrario se conserva en éste. Para aclarar, se elige el segundo valor ya que el mínimo de elementos en un grupo planteado es de 4. El umbral para la discriminación de valores atípicos se eligió experimentalmente (ver capítulo 6).

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Experimentos y Resultados

Se realizaron algunos experimentos para determinar los valores de los parámetros clave del sistema, y validar el uso de dichos valores. Para tener una idea de qué tan bien funciona el sistema para los parámetros evaluados es preciso definir una medida de desempeño.

6.1. Medida de desempeño

Al terminar la etapa de agrupamiento y eliminación de outliers, se obtiene una secuencia ordenada de elementos donde cada componente toma valores entre 0 y 4. Cada número corresponde a un grupo (cluster), de modo que, en el caso esperado, cada grupo se corresponde con una misma palabra repetida en la señal de audio. Estos números no están, *a priori*, asociados a un elemento específico del alfabeto del test RAN. Es decir, por ejemplo, en un test RAN de colores, no se conoce si el "0" corresponde a "azul", "rojo", "blanco", "verde", o "negro".

Para definir una medida de desempeño, es necesario primero asociar cada número (cluster) al elemento del patrón RAN al que se corresponde. Con este fin se toman todas las posibles combinaciones de asociaciones entre clusters y elementos del alfabeto. Luego se calcula la distancia de edición de convertir cada una de las combinaciones en la secuencia *ground truth*. Por último, se elige la combinación cuya distancia es menor que las demás. En definitiva, se obtiene una secuencia de palabras de la forma:

$$palabra_1, palabra_2, palabra_3, .., palabra_n$$

donde cada $palabra_i$ es una instancia de los posibles símbolos del alfabeto (vocabulario de palabras en el test RAN).

Cuando se calcula la distancia de edición entre la secuencia producto del procesamiento, y la secuencia patrón, es preciso definir los costos de que la secuencia tenga un elemento de más, de que tenga un elemento faltante, y el costo de cambiar un elemento por otro(como se vio en 4.2).

Claramente pronunciar una palabra fuera de lugar, saltear una palabra, o confundir una palabra por otra, son todos errores que deben ser contemplados en la

medida de desempeño. Se decidió asignar a cada posible error con el mismo valor $ci = cd = cr = 1$ y penalizar cada tipo de error por igual.

En particular, el costo de que la secuencia reconocida tenga un objeto de más no debería ser muy alto. Esta decisión se toma considerando que la aparición de una palabra de más no es una posibilidad esperable en la realización de un test RAN, y esta situación puede ser disparada por otros eventos difíciles de controlar. Se pretende tomar medidas previas contra estos eventos con la detección y eliminación de los valores atípicos.

6.2. Entrenamiento y validación del clasificador

El método de evaluación del clasificador[1], como ya fue mencionado, utiliza la distancia de edición entre la secuencia obtenida y la secuencia patrón del test. Esta medida tiene como resultado un número entero positivo igual al número de ediciones necesarias para convertir una secuencia en la otra. Para obtener una medida de desempeño (DES) en porcentajes se realiza la siguiente definición:

$$DES = \frac{l - d_{edicion}}{l} \times 100\,\%$$

donde l es el largo de la secuencia de referencia propuesta por el test RAN.

Como se menciono en el capítulo 2, la evaluación del sistema total se separa en dos partes: una para el segmentador automático y otra para el resto del sistema, o clasificador. La última parte mencionada incluye los procesos de extracción de características, cálculo de medida de distancia entre palabras, y agrupamiento. Nuevamente, al igual que en el segmentador automático, se decidió utilizar la validación cruzada como forma de entrenamiento y validación.

La base de datos se compone de 6 grabaciones realizadas por cada uno de los 20 hablantes. La división de los datos en conjunto de entrenamiento y conjunto de prueba se realiza tomando las grabaciones de 18 y 2 hablantes, para los respectivos conjuntos. Los parámetros a variar en el entrenamiento fueron:

1. F_{DTW}, factor de peso en la construcción de la matriz de distancia acumulada.
2. N_{MFCCs}, cantidad de MFCCs a extraer de una ventana de análisis.
3. σ, usado en la construcción de la matriz de afinidad W para realizar Spectral Clustering.
4. u_{va}, umbral que discrimina a los valores atípicos para su posterior eliminación.

A su vez se tomaron los siguientes rangos de variación para los parámetros en la validación cruzada:

[1]En el capítulo 2 se define el "clasificador" como todo el procesamiento del sistema total que sigue a la etapa de segmentación

HABLANTES	TRAINING	TESTING	LISTA 1	LISTA 2	LISTA 3	LISTA 4	LISTA 5	LISTA 6	PEOR LISTA
0 y 1	98.33 %	94.17 %	98.33 %	90.0 %	98.33 %	86.67 %	96.67 %	95.0 %	86.67 %
2 y 3	97.25 %	97.50 %	96.67 %	95.0 %	100 %	100 %	95.0 %	98.33 %	95.0 %
4 y 5	97.04 %	98.33 %	100 %	93.33 %	100 %	100 %	96.33 %	100 %	93.33 %
6 y 7	96.88 %	98.06 %	100 %	95.0 %	100 %	100 %	93.33 %	100 %	93.33 %
8 y 9	96.98 %	97.22 %	100 %	91.67 %	100 %	100 %	91.66 %	100 %	91.66 %
10 y 11	96.83 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %	100 %
12 y 13	97.06 %	98.33 %	100 %	93.33 %	100 %	100 %	96.67 %	100 %	93.33 %
14 y 15	97.04 %	98.33 %	98.33 %	96.67 %	98.33 %	98.33 %	98.33 %	100 %	96.67 %
16 y 17	96.88 %	98.06 %	100 %	96.67 %	100 %	100 %	98.33 %	100 %	96.67 %
18 y 19	97.38 %	97.50 %	100 %	90.0 %	100 %	100 %	96.67 %	98.33 %	90.0 %

Tabla 6.1: Resultados de validación cruzada para valores óptimos

1. $F_{DTW} \in$ [1,0 1,2 1,4 1,6 1,8]
2. $N_{MFCCs} \in$ [10 13]
3. $\sigma \in$ [6 7 8 9 10 11 12]
4. $u_{va} \in$ [13 14 15 16 17 18 19]

La implementación de Spectral Clustering utilizada corre el algoritmo de *K-means* n_{init} veces y toma para cada una de las corridas distintos estados iniciales para los centros de cada grupo [1]. Luego se elige el mejor valor obtenido dependiendo de la inercia del agrupamiento. La inercia es la suma de las distancias de cada muestra al centro del grupo correspondiente. El valor por defecto en la implementación es $n_{init} = 10$ y no se cambió. Esta inicialización aleatoria de los estados iniciales genera que los resultados obtenidos en una corrida de la prueba y en otra pueden dar resultados distintos. No obstante, se puede asumir que la diferencia no es significativa considerando que el algoritmo de *K-means* ya tiene bastante robustez con ese número de inicializaciones para los centros de los grupos.

Utilizando validación cruzada, se obtuvo un desempeño promedio de 97,75 % sobre la base datos y en la tabla 6.1 se muestran todos los resultados obtenidos. Los valores óptimos en el entrenamiento usando validación cruzada fueron siempre los mismos:

1. $F_{DTW} = 1{,}0$
2. $N_{MFCCs} = 10$
3. $\sigma = 7$
4. $u_{va} = 17$

Esto sugiere que los valores que dan mejores resultados no dependen de la selección de los 2 hablantes para el grupo de validación. Por lo tanto, estos parámetros son la mejor elección para el clasificador de acuerdo con los resultados.

En la tabla 6.1 se puede observar el máximo desempeño en el entrenamiento para cada grupo de la VC, el cual se obtiene utilizando los valores óptimos de los parámetros mencionados. Junto a cada uno de estos valores se encuentra el

desempeño para el conjunto de prueba correspondiente al grupo de la VC. El valor mínimo para el desempeño de los conjuntos de prueba es 94,17 % para el grupo 1 (hablantes 0 y 1) y el máximo es 100 % para el grupo 5 (hablantes 8 y 9). Además se encuentran los desempeños de los conjuntos de prueba para cada lista de palabras.

A primera vista se observa que en general los peores resultados se obtienen para la lista 2 y la lista 5. El desempeño promedio para estas listas es de 94,17 % y 96,29 % respectivamente. Estas listas son las que poseen valores atípicos en sus secuencias. Esto indica que el entrenamiento eligió un valor de umbral u_{va} más alto para favorecer los resultados en las listas sin valores atípicos. Considerando que los valores atípicos en las listas son escasos frente a la cantidad de elementos esperables en un test RAN, es razonable que el umbral u_{va} óptimo obtenido sea considerablemente mayor que el calculado en la sección 5.2 ($u = 14{,}62$).

Con respecto al valor óptimo obtenido para σ, los resultados cambian mínimamente si se utilizan los otros valores en el rango de variación elegido. Esto es lógico considerando que para notar cambios importantes σ tiene que variar exponencialmente. De pruebas de búsqueda previas ya se sabía que el punto de trabajo ideal se encontraba en torno al rango de valores utilizados para elegir σ.

Los experimentos realizados consumían mucho tiempo por lo tanto se probó solamente dos valores para el número de MFCCs. Esto significa que no se puede asegurar que el valor obtenido sea el óptimo. Sin embargo, en la mayoría de los trabajos que utilizan los MFCCs se trabaja con un número en el entorno de los dos planteados para el entrenamiento (10 y 13). La validación cruzada favoreció al menor de estos valores. Esto quizás se debe a que los coeficientes de orden mayor a 10 presentan mayor variabilidad y quizás su aporte termina perjudicando a la discriminación de palabras.

Para el parámetro F_{DTW}, como se vio en la sección 4.5, se esperaba que el valor de mejor desempeño sea mayor a 1,0 (para no favorecer la elección de avanzar en diagonal al construir el camino de alineamiento óptimo). Por este motivo, el rango de valores propuestos para este parámetro tenía a 1,0 en su extremo inferior. El mejor desempeño según la validación cruzada se dio para $F_{DTW} = 1{,}0$ pero dado lo anteriormente mencionado, decidimos realizar una prueba más para $F_{DTW} = 1{,}1$ (punto medio entre 1,0 y el siguiente valor de test, 1,2). El resto de los parámetros se dejaron en los valores determinados por la VC. Se probó 10 veces utilizando diferentes grupos de 2 hablantes como conjunto de prueba cada vez, al igual que en la parte de testing en la validación cruzada. Se obtuvo un desempeño promedio de 97,91 %, el cual es ligeramente mayor al obtenido utilizando $F_{DTW} = 1{,}0$. Esto pone en duda la elección de $F_{DTW} = 1,0$ como valor óptimo para este parámetro y sugiere que ese valor buscado se encuentra en un entorno próximo a estos valores evaluados. De todos modos, estas pequeñas variaciones no parecen tener un efecto significativo en el desempeño del sistema.

6.3. Evaluación del sistema total

Para concluir el trabajo con la base de datos, se decidió evaluar el sistema de punta a punta, el cual se compone del segmentador automático y el clasificador.

HABLANTES	DESEMPEÑO TEST
0 y 1	81.67 %
2 y 3	96.67 %
4 y 5	92.22 %
6 y 7	96.67 %
8 y 9	79.17 %
10 y 11	84.17 %
12 y 13	97.7 %
14 y 15	95.56 %
16 y 17	86.94 %
18 y 19	93.61 %
promedio	90.44 %

Tabla 6.2: Evaluación del sistema total

Previamente habíamos evaluado el desempeño de cada uno de los dos submódulos independientemente. Se utilizaron los parámetros óptimos encontrados previamente en los experimentos realizados para cada módulo independientemente. Se hizo la evaluación con todo la base de datos y se obtuvo un desempeño promedio igual al 90,44 %.

Por otro lado se hizo una evaluación siguiendo el mismo formato de conjuntos de prueba que en las validaciones cruzadas realizadas previamente. Es decir, se tomaron 10 conjuntos de prueba, cada uno conteniendo las grabaciones de 2 hablantes. Esto se hizo para analizar el efecto para cada conjunto de prueba de juntar el segmentador y el clasificador en un sistema completo.

Los resultados se pueden observar en la tabla 6.2. Es esperable que el peor desempeño (79,17 %) corresponda a los hablantes 8 y 9 ya que eran también los que daban peores resultados para el segmentador automático. El desempeño para los hablantes 0 y 1 (81,67 %) es una confirmación de que las fallas para cada módulo independiente producen un efecto incluso más negativo para el desempeño total del sistema. Para estos hablantes, el desempeño del segmentador es de 91,3 % y el del clasificador es de 94,17 %, los cuales son considerablemente mayores al desempeño del sistema total.

No obstante, los resultados en general son positivos considerando la complejidad del sistema total.

6.4. Evaluación para grabaciones reales de niños

Para finalizar, se decidió evaluar el sistema diseñado con grabaciones reales de niños realizando el test RAN. La evaluación fue hecha para cada módulo independientemente y no para el sistema total. Esto es debido a que los resultados para el segmentador, que se mostrarán a continuación, no fueron muy alentadores para proseguir a una evaluación del sistema total.

Las grabaciones reales disponibles no fueron realizadas tomando en cuenta

los requerimientos específicos de este sistema, y por tanto no cumplen con las hipótesis planteadas en 1.2 para el funcionamiento correcto del mismo. Se realizó una selección de 12 grabaciones buscando aquellas menos alejadas de las hipótesis. Para estas grabaciones se realizó el etiquetado manual de los tiempos de principio y fin de las palabras y a qué elemento corresponden. Hay 3 tests RAN presentes en las grabaciones: colores, números y objetos. El test RAN de objetos fue con el cual se creó nuestra base de datos. Hay 5 grabaciones dedicadas a objetos, 4 a colores y 3 a números. También teníamos disponible grabaciones de un test RAN cuyos elementos son letras, pero éstas fueron descartados ya que se alejaban aún más de las hipótesis planteadas para el funcionamiento correcto del sistema.

La cantidad de grabaciones disponibles en este conjunto no es suficientemente grande como para considerarlo un conjunto de prueba bueno. No obstante, sirve como prueba de concepto para tener una idea del desempeño del sistema para las grabaciones reales.

Grabación	Test RAN	Desempeño
1	COLORES	33.33 %
2	COLORES	30.0 %
3	COLORES	30.0 %
4	COLORES	81.81 %
5	NÚMEROS	34.48 %
6	NÚMEROS	60.0 %
7	NÚMEROS	13.3 %
8	OBJETOS	36.67 %
9	OBJETOS	23.33 %
10	OBJETOS	20.0 %
11	OBJETOS	16.67 %
12	OBJETOS	41.93 %

Tabla 6.3: Desempeño del segmentador automático para grabaciones reales utilizando los parámetros óptimos encontrados en el entrenamiento

Los resultados para el segmentador automático utilizando los parámetros óptimos encontrados en las etapas de entrenamiento se observan en la tabla 6.3. El desempeño promedio de todas las grabaciones fue de 35,13 %.

Para el clasificador utilizando la segmentación proporcionada en la base de datos, el desempeño promedio obtenido utilizando los parámetros óptimos encontrados previamente fue de 56,39 %. Los resultados para cada grabación se observan en la tabla 6.4.

El desempeño fue malo para ambos submódulos aunque es esperable debido a que los audios de los niños se alejan de las hipótesis para las grabaciones que se encuentran en la base de datos con las cuales se entrenaron al segmentador y al clasificador.

Grabación	Test RAN	Desempeño
1	COLORES	36.67 %
2	COLORES	50.0 %
3	COLORES	46.67 %
4	COLORES	53.33 %
5	NÚMEROS	53.33 %
6	NÚMEROS	50.0 %
7	NÚMEROS	66.67 %
8	OBJETOS	63.33 %
9	OBJETOS	53.33 %
10	OBJETOS	46.67 %
11	OBJETOS	100 %
12	OBJETOS	56.66 %

Tabla 6.4: Desempeño del clasificador para grabaciones reales utilizando los parámetros óptimos encontrados en el entrenamiento

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones

7.1. Conclusiones generales

Se desarrolló un sistema que logra automatizar la evaluación de un test RAN, con resultados auspiciosos. Esta afirmación se basa concretamente, en los resultados obtenidos en las pruebas realizadas sobre la base de datos generada para realizar los experimentos. Si bien estos valores de desempeño se obtuvieron de la validación cruzada, la variabilidad nula de los parámetros respecto a la elección del conjunto de test, indica que el sistema es robusto frente a estos parámetros, y es de esperar que una validación sobre un conjunto de test independiente del conjunto de entrenamiento, resulte en una medida de desempeño similar.

Podemos afirmar entonces que el proyecto realizado cumple parcialmente los objetivos propuestos. Para aseverar que el sistema desarrollado cumple los objetivos iniciales haría falta realizar un nuevo entrenamiento y validación con una base de datos representativa del escenario objetivo (grabaciones de niños). Este trabajo no se realizó dado que la base de datos obtenida con grabaciones de niños no cumple las hipótesis en que se basa el sistema.

La prueba de concepto sobre un conjunto de grabaciones de test RAN reales arrojó resultados razonables para la etapa de clasificación. Considerando que estas grabaciones no cumplen fielmente con las hipótesis en que se basa el sistema el desempeño de 76.38 % es alentador. Incluso aparece una situación no anticipada: en ocasiones el mismo elemento es leído con variaciones significativas entre instancias de la misma grabación. Aún así el resultado es razonable.

Para el segmentador, en cambio, la prueba con grabaciones reales produce un desempeño regular. Esto es causado principalmente por el nivel de ruido de estas señales y la corta separación entre palabras en algunas de las grabaciones. El desempeño de esta etapa puede mejorar considerablemente al mejorar el entorno sonoro de la realización del test, y las instrucciones sobre cómo realizarlo.

Se encontró que el segmentador de palabras es el punto débil del sistema. Su desempeño varía según el modo de hablar particular de cada persona y a causa de ruido e interferencias en el momento de la grabación. Además una segmentación realizada con errores, perjudica el trabajo de la etapa posterior y arrastra ese error

teniendo un efecto muy negativo en el desempeño del sistema.

Tanto el problema objetivo, como la solución desarrollada, pueden ser tan complejos como se decida. Durante el avance de este proyecto, se encontró situaciones que se decidió simplificar. Por ejemplo, se consideró imponer la restricción de que no exista más de un hablante en las grabaciones de audio a analizar, pero quizás es posible identificar distintos hablantes y separarlos como parte del sistema de evaluación. En este sentido, es de la opinión de los autores que, el trabajo se desarrolla con un nivel de complejidad suficiente para obtener resultados razonablemente buenos y útiles en el caso de la aplicación en el campo.

Respecto a la incorporación de este sistema como parte de una aplicación lúdica a implementarse en las computadoras del Plan Ceibal, se entiende, como se mencionaba en el párrafo anterior, que están dadas las condiciones para su realización. Sin embargo, sería necesario considerar primero algunas posibles mejoras (ver 7.2).

La creación de la base de datos grabaciones de audio desarrollada fue clave para poder trabajar con un problema con hipótesis claras y simplificadas. Esto constituye un paso previo necesario a trabajar con señales reales. Esta base de datos será publicada y compartida, con el fin de facilitar su uso para fines educativos y de investigación. Generar una base de datos de audio de voz con etiquetas es un trabajo que insume un tiempo considerable, y creemos que la comunidad puede beneficiarse de este subproducto de este proyecto de fin de carrera. La base de datos puede descargarse a través del siguiente link: `https://drive.google.com/file/d/1BqU5qGxQxNFUAUDqaELsI1MXwf91q2l5/view?usp=sharing`

Es importante destacar también que este trabajo fue acompañado de un proceso de aprendizaje sobre como enfrentar un proyecto de esta envergadura y sobre cómo trabajar en equipo. Este proceso se caracterizó por una planificación pobre y una mala coordinación de los esfuerzos, en particular en las etapas inicial y media de esta realización. Esto tuvo sus consecuencias en la etapa final, donde a pesar de ciertas mejoras en estos aspectos, la proximidad de los plazos de finalización, y tareas no previstas en las primeras etapas, dificultaron significativamente el trabajo.

7.2. Mejoras a futuro

Como vimos, el eslabón más débil en la cadena del sistema es la segmentación. Por tanto el segmentador debe ser lo primero a mejorar.

En la prueba de concepto, el desempeño del segmentador fue regular (35,13 %). Estas grabaciones fueron adquiridas en un entorno más ruidoso, y tampoco se cumple la hipótesis de separación entre palabras en la lectura.

Una forma de atacar este problema podría ser proponer varios valores de umbral en el algoritmo, y que luego se elija el que produce el número de palabras segmentadas más cercano al número total de elementos del test (dato conocido).

Otra posibilidad podría ser observar el rango de amplitud de la señal a analizar, y determinar un umbral único para cada señal, con un valor de una fracción fija de la amplitud total. Luego se identifican todos los segmentos de audio cuya amplitud

se mantiene por arriba de ese umbral por una duración mínima fija a determinar en torno a los 15 milisegundos. De todos los candidatos identificados, se eligen los 30 que contienen los mayores picos de amplitud. Esto fue aplicado en [15] en las condiciones exactas del problema que buscamos resolver, dando resultados alentadores.

Otro enfoque para mejorar la segmentación podría ser utilizar otro algoritmo. Este método alternativo agrega la hipótesis de que todas las palabras en los test RAN son bisílabas, es decir, compuestas por dos sílabas. En caso de cumplirse esta hipótesis, se podría trabajar reconociendo sílabas y encontrando parejas cercanas de sílabas para establecer las posiciones de las palabras. Existe mucha bibliografía en reconocimiento de voz mediante sílabas [11], y consideramos que esta es una alternativa atractiva.

Estas estrategias no son necesariamente excluyentes, y podrían complementarse para solucionar el problema de segmentación. Teniendo resuelto este problema, tendría sentido agregar una nueva salida del sistema aportando datos de duración de las palabras, duración de los silencios entre palabras, y estadísticas que puedan ser de utilidad a los investigadores.

En la prueba de concepto se vio que para esas grabaciones, la modificación del valor del umbral para reconocer outliers produce una mejora muy significativa en el clasificador (de 56.38 % a 76.38 %). Una forma simple de mejorar el desempeño puede ser proponer varios valores de umbral en el algoritmo, y que luego se elija el que produce el mayor desempeño.

Como se adelantaba, otra posible mejora sería realizar un reconocimiento del hablante principal en la grabación de audio, y descartar las intervenciones de otros interlocutores. Esto permitiría levantar la hipótesis de un sólo hablante en las grabaciones, permitiendo alguna interferencia leve de otros hablantes. Una posible forma de realizar esta discriminación es calcular la frecuencia fundamental de la voz para toda la señal [5]. La frecuencia fundamental de la voz en niños a los 3 años es de alrededor de 318Hz. Al crecer la frecuencia fundamental baja (la voz se hace más grave). A los 7 años ya existe diferenciación entre niños y niñas, mostrando frecuencias de 268Hz para los primeros y 295 para las últimas [13]. En contraste, para adultos, los hombres suelen encontrarse en el rango de 120-146Hz, las mujeres en 227-197Hz.

Para desarrollar cualquier trabajo futuro sobre este sistema, es fundamental obtener una base de datos de grabaciones de test RAN con niños que cumplan con las hipótesis del problema.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Base de Datos

Montevideo, Uruguay Marzo 2018

Base de datos de Voces Uruguayas para tests de sistema de análisis de pruebas neuropsicológicas

Guzmán Chalupa, Gabriel De Cola

Facultad de Ingeniería, Universidad de la República, Uruguay

Resumen
El enfoque principal de este trabajo es crear una base de datos de voces uruguayas con el propósito de entrenar un sistema de reconocimiento de voz para analizar grabaciones de pruebas psicológicas llamadas Denominación Automatizada Rápida(RAN, por sus siglas en inglés). Se utilizaron teléfonos móviles y la aplicación Whatsapp para realizar las grabaciones, en lugares con ruido ambiente bajo. La base de datos consta de un total de 120 grabaciones, para las cuales se etiquetó los fragmentos de audio con voz hablada. Se tomó información de edad y sexo de los hablantes.

Introducción
Esta base de datos se creó como parte del proyecto de fin de carrera "Reconocimiento automático de voz hablada para pruebas neuropsicológicas y detección de dificultades en el lenguaje" de Facultad de Ingeniería, UDELAR, en colaboración con el Centro de Investigación Básica en Psicología (CIBPsi), Facultad de Psicología, UDELAR.

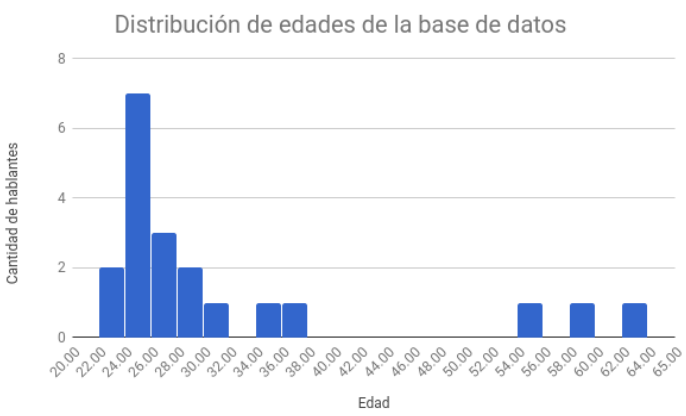
El test de Denominación Automatizada Rápida(RAN) evalúa la velocidad de acceso a la etiqueta léxica y ha probado ser útil como posible predictor de dificultades en la lectura, particularmente a edades tempranas. El test consiste en leer y nombrar en voz alta cada símbolo presente en una matriz. La lectura se realiza en el orden convencional de lectura, es decir, de izquierda a derecha y de arriba hacia abajo. Los símbolos a leer pueden ser letras, números, objetos, o colores, y pertenecerán a un alfabeto reducido, donde cada elemento se repetirá varias veces. Luego de realizado el test, se analizan las grabaciones para registrar el tiempo de lectura, y la cantidad de errores que pueden o no haberse cometido.

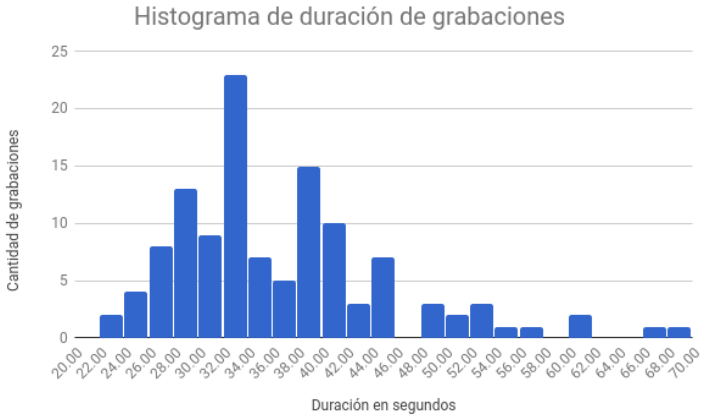
En el marco del proyecto de fin de carrera, nos proponemos desarrollar un sistema que permita realizar automáticamente el análisis de las grabaciones producidas al realizar el test en niños. Como primera instancia de evaluación del sistema, y ante dificultades de armar una base de datos con niños en los plazos establecidos, armamos esta base de datos con hablantes adultos.

Características

- Un total de 120 grabaciones y sus correspondientes etiquetas.
- 20 hablantes, 13 hombres y 7 mujeres.
- Edades entre 23 y 63 años, distribuidas como se puede ver en el histograma a continuación.
- Por cada hablante hay 6 grabaciones. De estas 6, 2 grabaciones representan a la lectura correcta del test RAN, mientras que las otras 4 son variaciones que representan diferentes desviaciones de la lectura correcta. Al final de este documento se puede ver las listas de palabras leídas por cada hablante.
- Entre 28 y 35 palabras por grabación.
- Frecuencia de muestreo de 16kHz.
- La duración media de las grabaciones es de 36 segundos. Al final de este documento se puede ver un histograma con la distribución de la duración de las grabaciones.

- Todos los audios se grabaron con la aplicación de mensajería Whatsapp en formato ogg y luego fueron convertidos a wav utilizando la biblioteca de herramientas de audio open-source Essentia .
- Las etiquetas se presentan en formato .csv





La base de datos se presenta en 20 carpetas, cada una de las cuales contiene las grabaciones que corresponden a un mismo hablante, así como los archivos de etiquetas correspondientes. El nombre de los archivos respeta el formato RxxHyySzz.wav, donde:

- xx toma un valor entre 01 y 06, de modo que Rxx corresponde a la lista de palabras xx.
- yy toma valores entre 00 y 19, así Hyy identifica al hablante.
- S representa al sexo del hablante, y puede ser F para mujeres y M para hombres.
- zz es un número que representa la edad del hablante.

Los archivos de etiquetas tienen el nombre RxxHyySzz_labels.csv y respetan el formato csv utilizado en CIBPsi. En estos archivos está disponible, para cada segmento de silencio y de voz hablada:

- Label. Si el campo toma el valor “word”, indica que corresponde a un segmento de voz hablada. Si el campo está vacío indica silencio.
- Start. Comienzo de segmento de audio.
- Mid. Marca el punto medio del segmento de audio correspondiente a esa palabra/silencio.
- End. Indica el fin del segmento.
- Duration. Marca la duración del segmento.

Esta información comienza en la quinta fila y tercer columna, como se puede ver en la siguiente imagen. Las filas y columnas precedentes respetan el formato con el que se trabaja en Facultad de Psicología en el estudio de estos tests.

Montevideo, Uruguay Marzo 2018

	A	B	C	D	E	F	G
1	TextGrid to CSV (D. Gibbon, 2008-11-23)						
2	Open the file with OpenOffice.org Calc or MS-Excel.						
3	Note that the timestamps are in milliseconds.						
4	TierType	TierName	Label	Start	Mid	End	Duration
5	IntervalTier	objetos		0	67.5	135	135
6	IntervalTier	objetos	word	135	457.5	780	645
7	IntervalTier	objetos		780	1140	1500	720
8	IntervalTier	objetos	word	1500	1730	1960	460
9	IntervalTier	objetos		1960	2220	2480	520
10	IntervalTier	objetos	word	2480	2776.75	3073.5	593.5
11	IntervalTier	objetos		3073.5	3456.75	3840	766.5
12	IntervalTier	objetos	word	3840	4070	4300	460
13	IntervalTier	objetos		4300	4710	5120	820

Listas de palabras

Lista 1	Lista 2	Lista 3	Lista 4	Lista 5	Lista 6
gato	gato	gato	gato	gato	gato
jugo	jugo	jugo	jugo	jugo	jugo
gato	gato	jugo	gato	gato	gato
mano	esteeee...	gato	silla	mano	mano
silla	mano	mano	queso	silla	silla
queso	silla	silla	silla	queso	queso
silla	queso	queso	gato	silla	silla
gato	silla	queso	mano	gato	gato
mano	gato	silla	queso	mano	mano
queso	mano	gato	jugo	ratón	queso
jugo	eehhhmmm...	mano	silla	jugo	jugo
silla	queso	mano	jugo	silla	silla
jugo	jugo	queso	mano	jugo	jugo
mano	silla	jugo	queso	mano	mano
queso	jugo	silla	mano	queso	queso
mano	mano	jugo	gato	mano	mano
gato	queso	mano	mano	gato	gato
silla	mano	queso	silla	silla	silla
mano	ratón	mano	gato	mano	mano
silla	gato	mano	queso	silla	silla
gato	silla	gato	jugo	queso	gato
queso	mano	silla	queso	jugo	queso
jugo	silla	mano	jugo	esteee...	jugo
queso	gato	silla	queso	queso	queso
jugo	queso	gato	silla	jugo	jugo
queso	jugo	queso	gato	queso	queso
silla	queso	jugo	mano	silla	silla
gato	jugo	jugo	jugo	gato	gato
mano	queso	queso		mano	mano

Montevideo, Uruguay Marzo 2018

jugo	perdón	jugo		jugo	jugo
	silla	queso			
	gato	silla			
	mano	gato			
	jugo	mano			
		jugo			

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Ejemplos de programación dinámica y principio de optimalidad

B.1. Problema de la mochila

Un clásico ejemplo de programación dinámica es la solución del problema de la mochila (Knapsack Problem).

Se nos dan n objetos y una mochila. El objeto i tiene un peso w_i y la mochila tiene una capacidad m. Si una fracción $x_i, 0 \leq x_i \leq 1$, del objeto i es colocado en la mochila, entonces se gana un beneficio $p_i x_i$. El objetivo es llenar la mochila de forma tal que se maximice el beneficio total obtenido y sin exceder la capacidad de la mochila ($\sum x_i w_i \leq m$).

La solución a este problema puede ser vista como el resultado de una secuencia de decisiones. Debemos decidir los valores de los x_i. Primero podemos tomar una decisión sobre el valor de x_1, luego x_2, luego x_3, y así sucesivamente. Una secuencia de decisiones óptima maximiza la función objetivo $\sum p_i x_i$, y satisface la restricción $\sum w_i x_i \leq m$.

En particular, si ordenamos los n objetos en orden decreciente de relación de beneficio por unidad de peso (p_i/w_i), de modo que $p_i/w_i \geq p_{i+1}/w_{i+1}$, podemos decidir $x_i = 1$ para todo objeto i mientras se cumpla $\sum w_i x_i \leq m$. Habrá un último objeto j cuyo valor x_j será no nulo, $0 \leq x_j \leq 1$, a partir del cual todos los x_i con $i > j$ serán nulos. Se puede demostar que la secuencia de decisiones obtenida será óptima, y por tanto se puede hallar tomando una secuencia de decisiones y sin necesidad de calcular ninguna otra secuencia posible.

En el problema de la mochila, se puede llegar a la secuencia de decisiones óptima sin necesidad de calcular secuencias no óptimas.

B.2. Camino más corto

Supongamos que queremos encontrar el camino más corto entre dos ciudades lejanas. Para llegar a destino se pueden elegir distintos caminos de diferente largo, dependiendo de las ciudades intermedias por las que decidamos pasar. Una forma

de encontrar el camino más corto desde la ciudad i a la ciudad j es decidir primero qué ciudad será la segunda, cual será la tercera, y así sucesivamente. Una secuencia de decisiones óptima es aquella que resulta en el camino más corto. Si llamamos A_i al conjunto de las ciudades conectadas directamente a i, ¿Cual de todas las ciudades de A_i debería ser la segunda ciudad para obtener el camino más corto? No se puede elegir una segunda ciudad, y garantizar que las decisiones futuras producirán una secuencia óptima. En este problema es necesario calcular secuencias no óptimas para determinar la secuencia óptima.

Una opción posible para resolver este problema es calcular todos los caminos posibles y luego elegir el más corto. Podemos evitar calcular todos los caminos si utilizamos de forma conveniente el Principio de Optimalidad.

B.3. Principio de optimalidad

El principio de optimalidad afirma que una secuencia de decisiones óptima tiene la propiedad de que para cualquier estado inicial y cualquier decisión, las decisiones siguientes deben constituir una secuencia de decisiones óptimas con respecto al estado resultante de la primer decisión.

Por ejemplo, consideremos el problema de encontrar el camino más corto entre dos ciudades. Sea $i, i_1, i_2, ..., i_k, j$ el camino más corto desde i a j. Comenzando en i, se tomó una decisión para llegar a i_1. Luego de tomar esta decisión, el estado del problema es definido por i_1, y necesitamos encontrar un camino de i_1 a j. Está claro que la secuencia $i_1, i_2, ..., i_k, j$ debe constituir el camino más corto de i_1 a j. Supongamos que esto no es así, y $i_1, r_1, r_2, ..., r_q, j$ es el camino más corto de i_1 a j. Entonces $i, i_1, r_1, ..., r_q, j$ es un camino de i a j más corto que el camino $i, i_1, i_2, ..., i_k, j$. Este resultado absurdo implica que el principio de optimalidad aplica para este problema.

Dado el conjunto A_i de las ciudades adyacentes a i, para cada k perteneciente a A_i, sea Γ_k el camino más corto de k a j. Por tanto, el camino más corto de i a j es el más corto de los caminos $\{i, \Gamma_k | k \in A_i\}$. Este tipo de razonamiento utilizando el principio de optimalidad permite hallar el camino más corto sin tener que calcular todas las posibles secuencias de decisiones.

Apéndice C

Curva ROC

Una Curva ROC (acrónimo de Receiver Operating Characteristic, o Característica Operativa del Receptor) es una representación gráfica del desempeño de un sistema de clasificación binario variando un parámetro de entrada al sistema que generalmente representa un umbral de decisión.

En una clasificación de dos clases los tipos de aciertos y errores se pueden dividir en cuatro casos: Verdaderos Positivos (en inglés True Positives y su acrónimo TP), Falsos Positivos (en inglés False Positives y su acrónimo FP), Verdaderos Negativos (en inglés True Negatives y su acrónimo TN) y Falsos Negativos (en inglés False Negatives y su acrónimo FN). Las instancias positivas (P) son la suma de los TP y los FN, mientras que las instancias negativas (N) son la suma de los FP y los TN.

La Curva ROC representa la relación directa entre la Sensibilidad o Razón de Verdaderos Positivos (en inglés True Positives Rate y su acrónimo TPR) y la Razón de Falsos Positivos (en inglés False Positives Rate y su acrónimo FPR) según la variabilidad del umbral de decisión.

El TPR se define como la razón entre los Verdaderos Positivos (TP) y las instancias positivas (P) y el FPR se define como la razón entre los Falsos Positivos y las instancias negativas (N), o sea:

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$$

Usualmente se tiene que reducir los FP implica aumentar los FN, y viceversa. Por tanto la curva ROC es una herramienta que sirve para elegir modelos óptimos y analizar la relación costo beneficio que implica tomar una decisión. Por ejemplo, si hacer un diagnóstico positivo erróneo es menos costoso que dar un diagnóstico negativo erróneo, conviene elegir un umbral que minimice lo más posible (considerando la relación costo beneficio) la cantidad de diagnósticos negativos erróneos aunque esto implique aumentar la cantidad de diagnósticos positivos erróneos.

Un espacio ROC se define por los FP y TP como ejes x e y respectivamente,

y representa los intercambios entre verdaderos positivos y falsos positivos. Cada resultado de predicción representa un punto en el espacio ROC.

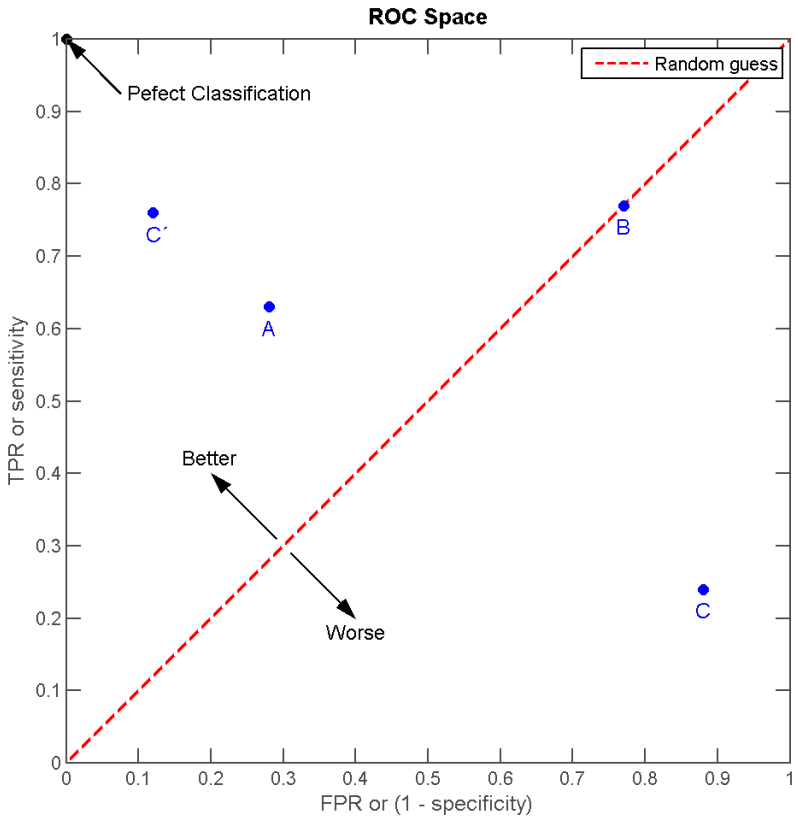


Figura C.1: Espacio ROC (imagen extraída de [22])

En la figura C.1 se puede apreciar un espacio ROC. El mejor método posible de predicción se situaría en un punto en la esquina superior izquierda, o coordenada (0,1) del espacio ROC, representando un clasificación perfecta. Por el contrario, una clasificación totalmente aleatoria daría un punto a lo largo de la línea diagonal (como el punto B), que se llama también línea de no-discriminación, desde el extremo inferior izquierdo hasta la esquina superior derecha (independientemente de los tipos de base positivas y negativas). Un ejemplo típico de adivinación aleatoria sería decidir a partir de los resultados de lanzar una moneda al aire, a medida que el tamaño de la muestra aumenta, el punto de un clasificador aleatorio de ROC se desplazará hacia la posición (0.5, 0.5).

La diagonal divide el espacio ROC. Los puntos que están por encima de la diagonal como A y $C*$, representan los buenos resultados de clasificación (mejor que el azar), puntos por debajo de la línea de los resultados pobres como C (peor que al azar). Sin embargo se puede invertir los resultados del método de predicción

para el punto C para obtener un buen predictor como lo es el C^*. Por lo tanto el peor resultado de predicción que se puede tener es uno en la diagonal (por ejemplo el punto B) ya que no aporta nada en el poder de decisión.

Como ya se mencionó la curva ROC se puede usar para generar estadísticos que resumen el rendimiento (o la efectividad, en su más amplio sentido) del clasificador. El indicador más utilizado en muchos contextos es el área bajo la curva ROC o AUC. Este índice se puede interpretar como la probabilidad de que un clasificador puntuará una instancia positiva elegida aleatoriamente más alta que una negativa.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

DFT

La Transformada de Fourier (TF) es una función matemática que puede ser usada para mostrar diferentes partes de una señal continua. Se utiliza mayormente para convertir una señal en el dominio del tiempo a una señal en el dominio de la frecuencia [21]. La TF muestra qué frecuencias están presentes en una señal. Por ejemplo, al considerar una señal de audio con tres notas musicales, al graficar la TF (con la frecuencia en el eje de las x, y la intensidad en el eje de las y) se verá un pico en cada frecuencia correspondiente a las notas musicales presentes en la señal.

La Transformada de Fourier Discreta(DFT) es un análisis en el dominio de frecuencia para una secuencia de números de duración finita. La función transforma la secuencia en otra con valores complejos y del mismo largo [16]. Mantiene las virtudes de la TF: revela elementos periódicos de la señal y muestra la fuerza relativa de estos elementos. Dada una señal en tiempo discreto x_n con N muestras, su transformada X_n está dada por:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}$$

En la práctica, el costo computacional del cálculo de la DFT se reduce utilizando la Transformada Rápida de Fourier (FFT).

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

DCT

La Transformada de Coseno Discreta, a diferencia de la DFT, expresa una secuencia de números finita en términos de una suma de cosenos a distintas frecuencias y la representación de una secuencia real mediante esta función es también una secuencia real [16]. Hay muchas variantes de esta transformada pero la más común es la DCT-II, la cual es denominada simplemente como "DCT", y se define de la siguiente forma, siendo x_n una secuencia de largo N:

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}(n + \frac{1}{2}k)\right]$$

La DCT se utiliza en muchas aplicaciones de compresión de datos con preferencia sobre la DFT debido en parte a que se trabaja únicamente con números reales, pero principalmente debido una propiedad que se denomina generalmente "compactación de la energía". La DCT tiende a concentrar la mayor parte de la información de la señal en los coeficientes de baja frecuencia. Por lo tanto, se necesita un menor número de coeficientes para representarla.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice F

Validación Cruzada

La validación cruzada es un proceso que crea una división en conjuntos de entrenamiento y de prueba a partir de un solo conjunto de datos [4]. Los datos se reparten en k subconjuntos (S_1, S_2, ..., S_k). Estos subconjuntos son aproximadamente del mismo tamaño. El algoritmo de aprendizaje luego se aplica k veces, para $i = 1$ a k, donde cada vez se usa la unión de todos los subconjuntos que no sean S_i como conjunto de entrenamiento y S_i se usa como conjunto de prueba.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] sklearn.cluster.spectralclustering. [Web; accedido el 26-04-2018].

[2] Ladan Baghai-Ravary and Steve Beet. Automatic speech signal analysis for clinical diagnosis and assessment of speech disorders, 2013.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, O. Mayor, Gerard Roma, Justin Salamon, J. R. Zapata, and Xavier Serra. Essentia: an open-source library for sound and music analysis. In *ACM International Conference on Multimedia (MM'13)*, pages 855–858, Barcelona, Spain, 21/10/2013 2013.

[4] Geoffrey I. Webb Claude Sammut. *Encyclopedia of Machine Learning*. Springer, 2011.

[5] Carlos Arturo García Gómez. Identificación del hablante empleando cepstro y curva melódica. 2015.

[6] J. Shi, J.Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 2(8):889–905, 2000.

[7] Dr. Ing. José Joskowicz. Codificación de voz y video. Technical report, 2017.

[8] Biing-Hwang Juang, L. Rabiner, and J. Wilpon. On the use of bandpass liftering in speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(7):947–954, Jul 1987.

[9] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

[10] M. M. Homayounpour M. H. Moattar. A simple but efficient real-time voice activity detection algorithm. *17th European Signal Processing Conference*, 2009.

[11] H. Meneido and J. P. Neto. The use of syllable segmentation information in continuous speech recognition hybrid systems applied to the portuguese language. 2000.

[12] Federico Miyara. La voz humana.

[13] M. T. Molina Hurtado, S. Fernández González, F. Vázquez de la Iglesia, and A. Urra Barandiarán. Voz del niño. *Revista de Medicina Universidad de Navarra*, 50(3):31–43, 2006—.

[14] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[15] G. F. Neuhaus, C. D. Carlson, W. M. Jeng, Y. Post, and P. R. Swank. The reliability and validity of rapid automatized naming scoring software ratings for the determination of pause and articulation component durations. *Educational and Psychological Measurement*, 61(3):490–504, 2001—.

[16] T. F. Quatieri. *Discrete-Time Speech Signal Processing: Principles and Practice*. 2002.

[17] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[18] Jyrki Rissanen. Naming game - an automated tool for analyzing and practicing rapid serial naming on dyslexic persons. *Computers Helping People with Special Needs*, (11):700–704, 2008.

[19] S.S. Stevens, J. Volkmann, and E.B. Newman. *A scale for the measurement of the psychological magnitude pitch*. 1937.

[20] Gabriela Saráchaga, Virginia Sartori, Laura Vignoli. Identificación automática de resumen en canciones, 2006.

[21] Wikipedia. Fourier transform, 2017. [Web; accedido el 14-04-2018].

[22] Wikipedia. Receiver operating characteristic, 2017. [Web; accedido el 28-04-2018].

[23] Carreiras M. Valle Lisboa J. Zugarramurdi C., Lallier M. Diseño de una evaluación digitalizada de predictores de las dificultades lectoras, 2016.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el lunes 24 septiembre, 2018.
http://iie.fing.edu.uy/

Sociedad de Ingeniería de Audio

Artículo de Congreso

Congreso Latinoamericano de la AES 2018
24 a 26 de Septiembre de 2018
Montevideo, Uruguay

Análisis Automático de Voz Hablada para Detección de Dificultades en el Aprendizaje de la Lectura

Gabriel De Cola,[1] Guzmán Chalupa,[1] Martín Rocamora[1] y Pablo Cancela[1]

[1] UDELAR FING, Instituto de Ingeniería Eléctrica
Montevideo, Montevideo, 11300, Uruguay

gabriel.de.cola, guzman.chalupa, rocamora, pcancela @fing.edu.uy

RESUMEN

Se propone un sistema para el análisis de un test de dificultad lectora en niños: la nominación automatizada rápida (RAN), que consiste en la denominación secuencial de un conjunto de 5 símbolos dispuestos en una matriz de 5 filas. La solución se basa en que los elementos del test se repiten. Se segmenta la señal de audio identificando inicio y fin de cada palabra, y se extraen características para compararlas, teniendo en cuenta deformaciones temporales. Las palabras se agrupan automáticamente y se compara la matriz original y la secuencia identificada, lo que permite detectar los errores cometidos.

0. INTRODUCCIÓN

El test RAN (del inglés, *Rapid Automatized Naming*) es un predictor de dificultades del lenguaje muy utilizado en niños [1], en el que se debe nombrar en voz alta y de forma secuencial una serie de elementos dispuestos en una matriz. Los elementos de la matriz provienen de un alfabeto de 5 símbolos, pudiendo ser letras, números, colores u objetos. La evaluación del test mide el tiempo en realizarlo y los errores cometidos.

En este trabajo se analizan de forma automática grabaciones de tests RAN, evaluando los errores de omisión, sustitución o inserción de elementos en la lectura de la secuencia. Se construyó una base de datos de archivos de audio bajo ciertas hipótesis: contener sólo la voz de quien realiza el test, con pausas entre palabras, y buena relación señal a ruido.

1. SOLUCIÓN PROPUESTA

La solución propuesta aprovecha la repetición de elementos en la matriz, es decir, se basa en la autosimilitud de la señal de audio. El sistema implementado segmenta la señal de audio para identificar el inicio y el fin de cada palabra usando un detector de actividad vocal (VAD). Luego extrae características que permiten comparar las palabras detectadas. Existe una etapa de alineamiento entre los segmentos de audio correspondientes a las palabras detectadas, que busca establecer su similitud, teniendo en cuenta deformaciones temporales entre realizaciones diferentes de una mis-

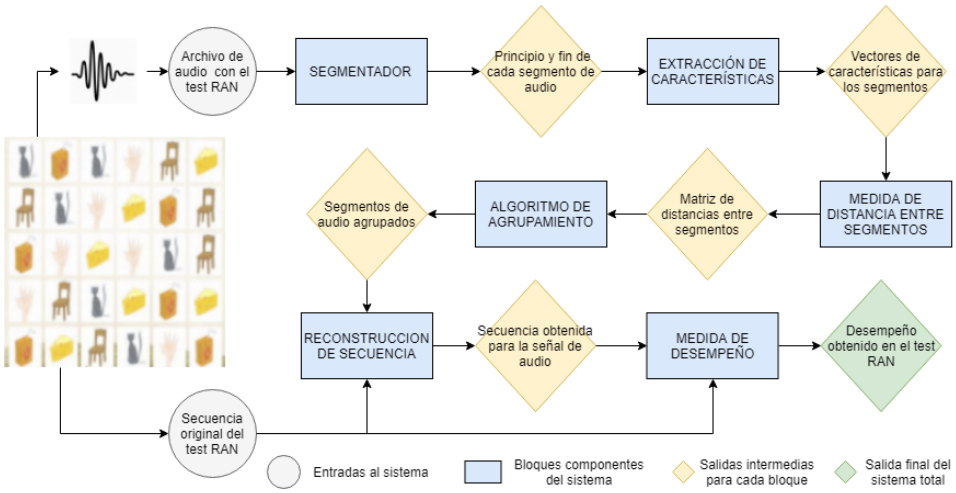


Figura 1: Diagrama de bloques del sistema propuesto.

ma palabra. Luego, los segmentos de audio se agrupan automáticamente para establecer cuáles corresponden a una misma palabra. Por último, se realiza una comparación entre la matriz original del test RAN y la secuencia de palabras identificadas, que permite detectar los errores cometidos y asignar un valor de desempeño. La Fig. 1 muestra un diagrama del sistema implementado.

El segmentador de palabras usa la energía en tiempo corto de cada trama de audio, y solo si supera cierto umbral se considera voz hablada [2]. Luego, se extraen los Coeficientes Cepstrales de Frecuencia Mel (MFCC) que representan características tímbricas de la señal de audio. Para comparar los segmentos de voz identificados se emplea Dynamic Time Warping (DTW), ya que permite tener en cuenta deformaciones temporales entre realizaciones diferentes de una misma palabra. La implementación de DTW usa la distancia cepstral ponderada por seno elevado entre los MFCCs de las tramas de audio. El camino de alineamiento óptimo entre dos palabras se obtiene calculando la distancia acumulada mínima entre las tramas de audio correspondientes.

Las palabras se agrupan usando Spectral Clustering de modo de obtener un grupo por cada tipo de símbolo del test. Para ello se construye una matriz de similitud entre cada pareja de segmentos de voz usando la distancia obtenida con DTW. Además, se detectan y eliminan valores atípicos comparando la distancia de cada elemento con el resto de los elementos del grupo.

Por último, para recrear la secuencia vocalizada se consideran todas las posibles correspondencias entre grupos y símbolos. Usando distancia de edición se comparan las secuencias resultantes con la secuencia RAN de referencia. El puntaje resultante se basa en los errores cometidos para la secuencia con menor distancia.

2. Base de datos

Se generó una base de datos de 120 grabaciones de audio, para una misma versión del test RAN cuyos símbolos son objetos, con etiquetas de inicio y fin de palabras, en la que participaron 20 hablantes adultos (hombres y mujeres). Para cada hablante hay dos grabaciones donde se lee correctamente la secuencia, y cuatro donde se introducen algunos errores de interés.

3. Experimentos y resultados

La base de datos creada permite evaluar el sistema propuesto. Se realizaron experimentos para entrenar y evaluar el desempeño del segmentador y el resto del sistema (denominado clasificador) de forma independiente, usando validación cruzada con una partición de los datos por hablante.

Para la evaluación del sistema total se utilizaron los parámetros óptimos obtenidos de entrenar cada módulo independientemente. El desempeño del segmentador es de 92.87 %. La clasificación con segmentación manual alcanza 97.75 % de acierto, mientras que el sistema total con segmentación automática obtiene 90.44 % de acierto.

4. Conclusiones

Se propuso un sistema para el análisis automático de la grabación de un test RAN basándose en la autosimilitud de la señal. Se creó una base de datos con adultos que arroja buenos resultados. Se puede esperar que al evaluar en niños el desempeño disminuya, no obstante, los resultados obtenidos son prometedores, validan el enfoque y sugieren lineas de trabajo futuro.

Agradecimientos

Trabajo parcialmente financiado por ICT4V.

Referencias

[1] Carreiras M. Valle Lisboa J. Zugarramurdi C., Lallier M., “Diseño de una evaluación digitalizada de predictores de las dificultades lectoras,” 2016.

[2] T. F. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*, 2002.

Análisis Automático de Voz Hablada para Detección de Dificultades en el Aprendizaje de la Lectura

Resumen

La nominación automatizada rápida (RAN), consiste en la denominación secuencial de los elementos de una matriz de 5 filas. Estos elementos pertenecen a un alfabeto de 5 símbolos. El desempeño de un niño en esta tarea es un buen predictor de futuras dificultades en la lectura.

Se propone un sistema para el análisis de grabaciones de niños realizando el test RAN. La solución propuesta se basa en que los elementos del test se repiten. Se segmenta la señal de audio identificando inicio y fin de cada palabra, y se extraen características para compararlas. Las palabras se agrupan automáticamente y se compara la matriz original y la secuencia identificada, lo que permite detectar los errores cometidos.

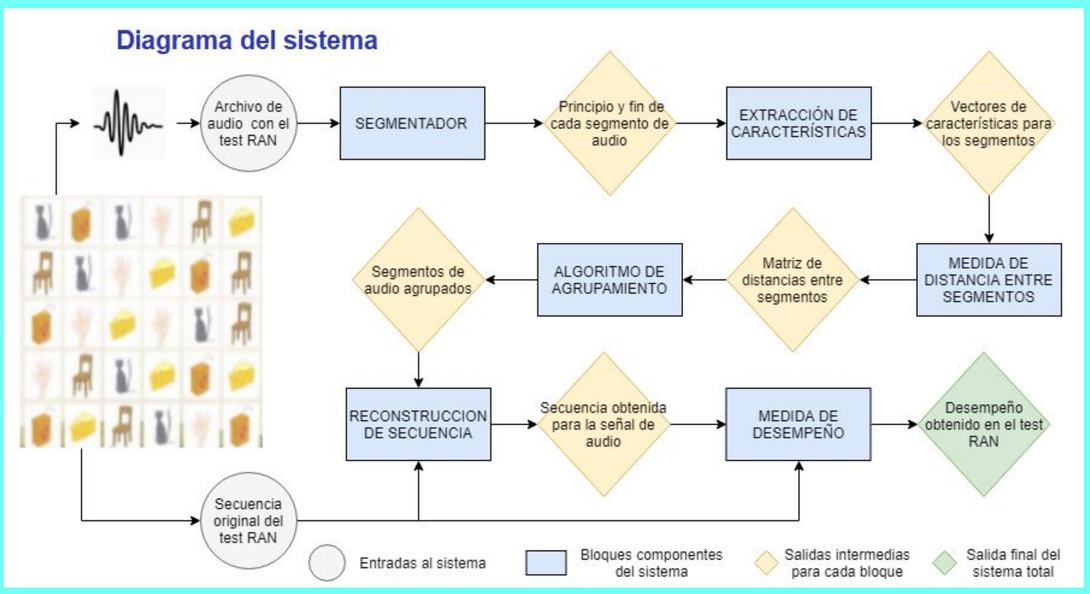
Hipótesis

Para el correcto funcionamiento del sistema planteado, las grabaciones a evaluar cumplen:

- Corresponden a una única persona en cada grabación: el niño evaluado.
- Existe silencio entre palabras.
- Las palabras son articuladas correctamente.
- Buena relación señal a ruido.

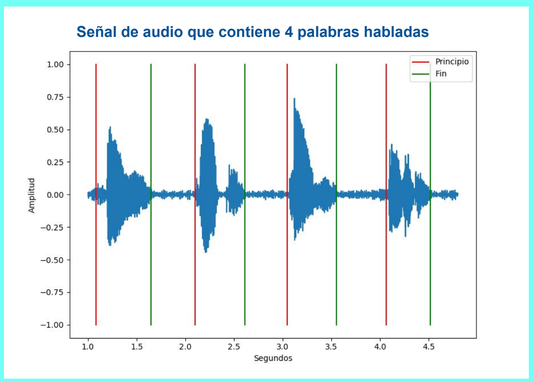
Base de datos

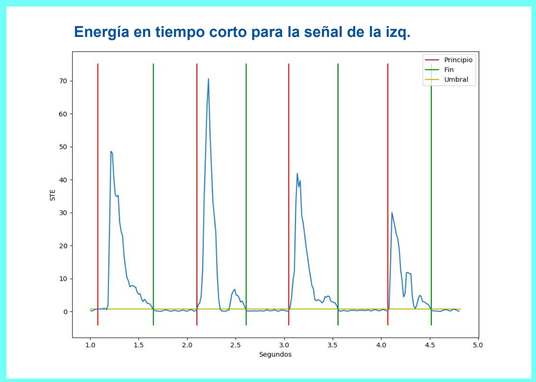
Para poder entrenar el sistema y evaluar su desempeño, se generó una base de datos de grabaciones de personas realizando la lectura del test RAN. Se contó con la colaboración de 20 voluntarios para obtener 120 simulaciones de un test RAN de cinco palabras, etiquetadas con principio y fin de cada palabra.



Segmentador

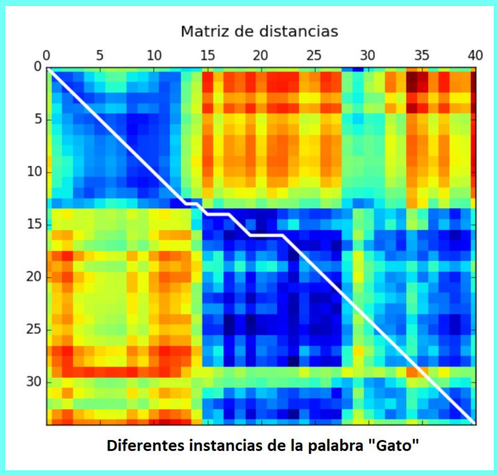
El segmentador de palabras usa la energía en tiempo corto de cada trama de audio, y sólo si supera cierto umbral se considera voz hablada. Además se establece un máximo de tramas con silencio para el cual dos segmentos de audio serían considerados como una sola palabra.

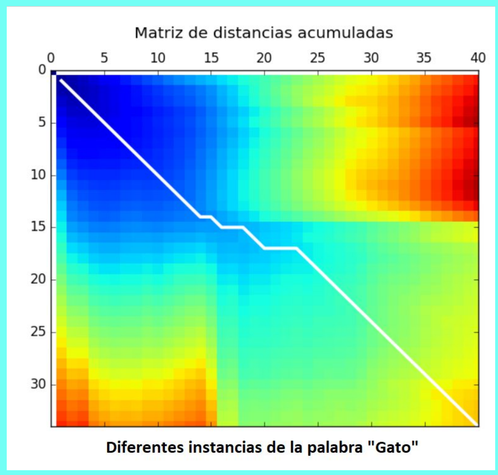




Medida de distancia entre segmentos de audio

Para cada segmento de audio de voz identificado por el segmentador de palabras, se extraen los Coeficientes Cepstrales de Frecuencia Mel (MFCC). Cada trama de audio se corresponde con un vector de características que lo describe. Para comparar los segmentos de voz se emplea Dynamic Time Warping (DTW), ya que permite tener en cuenta deformaciones temporales entre realizaciones diferentes de una misma palabra. Esta técnica busca el camino de alineamiento óptimo entre dos segmentos de audio al calcular la distancia acumulada mínima entre dos secuencias de tramas de audio.



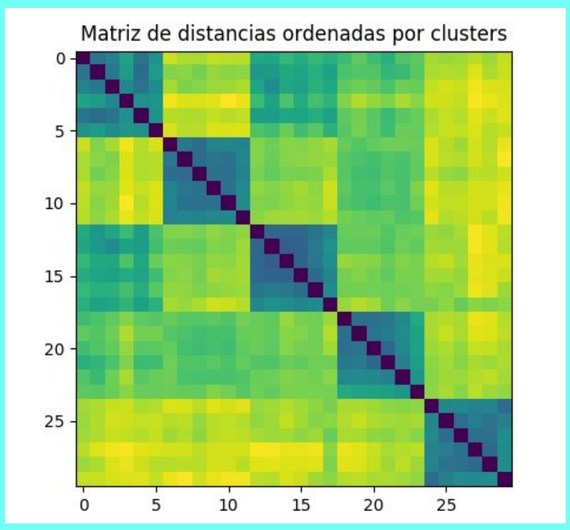


Agrupamiento

Las palabras se agrupan usando Spectral Clustering de modo de obtener un grupo por cada tipo de símbolo del test. Para esto se construye una matriz con la distancia dada por DTW entre cada pareja de segmentos. Además, se detectan y eliminan valores atípicos comparando la distancia de cada elemento con el resto de los elementos del grupo.

Reconstrucción

Para asociar cada grupo de segmentos de audio con un símbolo del test RAN se compara la secuencia original del test RAN con todas las posibles correspondencias grupo-símbolo utilizando distancia de edición. La secuencia reconstruida asociada al test será la que muestre la mínima distancia de edición.



Medida de desempeño

Como medida de desempeño se utiliza una normalización de la distancia de edición de la secuencia hallada con la secuencia de referencia.

Resultados

La base de datos creada permite evaluar el sistema propuesto. Se realizaron experimentos para entrenar y evaluar el desempeño del segmentador y el resto del sistema (denominado clasificador) de forma independiente, usando validación cruzada con una partición de los datos por hablante.
Los resultados se observan en la siguiente tabla:

Módulo	Desempeño
Segmentador	92.87%
Clasificador	97.75%
Sistema total	90.44%