UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de algoritmos para el aprendizaje automático de palabras en idiomas desconocidos a partir de audio

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: David Gómez Requena

Tutor: Emilio Sanchís Arnal

Cotutor: Fernando García Granada

Curso 2018-2019

Resumen

En este trabajo van a ser desarrollados algoritmos que sirven para simular el aprendizaje de las lenguas que realizan los niños. El proceso principal consiste en la búsqueda en discursos largos de audio, de segmentos de voz similares que puedan ser palabras de la lengua. Por tanto, sin tener ningún conocimiento de la misma, poder llegar a aprenderlas. Esto puede ser trasladado a cualquier lengua desconocida con pocos recursos lingüísticos. También sirve para detectar palabras nuevas, neologismos, nombres de personas o sociedades que no están previstos en los reconocedores de habla estándar, etc. Por lo tanto el objetivo principal del trabajo puede definirse como el desarrollo de algoritmos de Programación Dinámica eficientes que permitan comparar las muestras de audio entre sí.

Palabras clave: Descubrimiento de palabras; Algorítmica; Procesamiento del habla; Programación dinámica.

Abstract

During this project algorithms will be developed with the objective of simulate the process of language learning performed by childrens. The main process consists in the search in long audio speeches, of similar voice segments that can be words of the language. Therefore, without having any knowledge of it, to get to learn them. This can be transferred to any unknown language with few linguistic resources. It also serves to detect new words, neologisms, names of people or societies that are not provided for in standard speech recognizers, etc. Therefore, the main objective of the work can be defined as the development of efficient Dynamic Programming algorithms that allow audio samples to be compared with each other.

Keywords: Word searching process; Algorithmic; speech processing; Dynamic programming.

Resum

En aquest treball van a ser desenvolupats algorismes que serveixen per a simular l'aprenentatge de les llengües que realitzen els xiquets. El procés principal consisteix en la cerca en discursos llargs d'audio, de segments de veu similars que puguen ser paraules de la llengua. Per tant, sense tindre cap coneixement d'aquesta, poder arribar a aprendre-les. Això pot ser traslladat a qualsevol llengua desconeguda amb pocs recursos lingüístics. També serveix per a detectar paraules noves, neologismes, noms de persones o societats que no estan previstos en els reconeixedors de parla estàndard, etc. Per tant, l'objectiu principal del treball pot definir-se com el desenvolupament d'algorismes de programació dinàmica eficients que permeten comparar mostres d'àudio entre si.

Paraules clau: Descobriment de paraules; Algorítmica; Processament de la parla; Programació dinàmica.

Índice General

Índice de Figuras

Índice de Algoritmos

Capítulo 1
Introducción

El concepto principal entorno al que va a girar todo el proyecto es el procesamiento del habla. Se puede definir al procesamiento del habla o *"speech processing"* como el estudio de las señales del lenguaje y de los diferentes métodos para su procesamiento. También como la intersección del procesamiento de señales digitales y del procesamiento del lenguaje natural.

Este campo de investigación se compone de tres subapartados:

1. La adquisición del audio.
2. La manipulación y transformación de su información.
3. El retorno de dicho audio descompuesto en señales más sencillas que puedan ser procesadas.

En la actualidad es aplicado en sistemas interactivos de voz, asistentes virtuales, identificadores del habla, reconocedores de emociones, centros de distribución de llamadas y robótica en general.

Durante el proyecto no se va a entrar en detalle en los diferentes procesos de adquisición de audios. Se centrará la atención sobre todo, en la manipulación de la información de la que se dispone, de su transformación y de su procesamiento con el fin de obtener unos resultados visibles y evaluables.

1.1 Reconocimiento del habla

El reconocimiento automático del habla trata la extracción en texto del discurso de un locutor. Esta extracción se realiza de forma que se pueda obtener y procesar la información contenida en el audio independientemente de las características individuales de los hablantes, así como del propio idioma utilizado en mismo.

Este reconocimiento no conlleva la interpretación o entendimiento del significado del texto, esta tarea se realizaría posteriormente mediante técnicas de procesamiento del lenguaje natural. El objetivo principal es representar la entrada de audio en términos de palabras u otras unidades lingüísticas organizando así la información para la aplicación de futuros algoritmos.

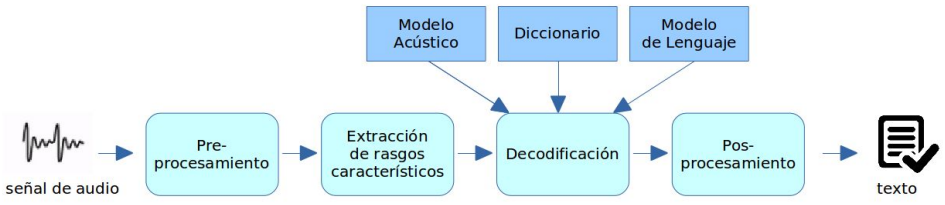

Figura 1. Sistema de reconocimiento del habla

La Figura 1 muestra un sistema de reconocimiento del habla típico con las diferentes fases por las que pasa un fichero de audio hasta que es convertido a un conjunto de vectores en un fichero de texto para su posprocesamiento.

1. La primera etapa es el preprocesamiento, durante esta etapa se graba la señal sonora y es discretizada a cierta frecuencia (16 o 8 kHz) con tal de normalizar la señal. También se aplican ciertos filtros opcionales con el objetivo de reducir el ruido de fondo, marcar las diferencias entre tipos de vocalización o distinguir los silencios.
2. La segunda etapa tiene como nombre la extracción de rasgos característicos. En esta fase la señal es dividida en ventanas de tiempo (*frames*), usualmente de 20 milisegundos cada una con superposición de 10 milisegundos entre ellas. A dichas señales acústicas se le aplican ciertas transformaciones matemáticas, como *Fourier* y coseno discreto, así como otra serie de filtros y procesos de normalización con el fin obtener un vector de coeficientes representativo de la señal *(MEL Cepstrum)*.
3. La tercera etapa corresponde a los procesos de decodificación, donde se centran la gran mayoría de algoritmos de reconocimiento del habla. Aquí se calcula la secuencia de palabras más probable de corresponder a la señal representada por los vectores de rasgos característicos. Podemos dividir este apartado a su vez en tres subetapas.
    a. Modelo Acústico. Hace referencia al formato de audio o sonido, asociado a cada fonema o palabra.
    b. Diccionario. Lista de palabras y fonemas que conforman el idioma o lenguaje que estamos analizando.
    c. Modelo de lenguaje. El cual está compuesto por probabilidades de palabras, secuencias de las mismas, correspondencias, etc.
4. Por último la etapa de postprocesamiento, donde no solamente se busca devolver la frase, fonema o palabra obtenida durante los pasos anteriores como resultado definitivo. En esta etapa se aplican nuevos ajustes empleando fuentes adicionales de información con el fin de conocer el contexto y tratar de hacer más precisa la hipótesis final.

Tener posesión de todos estos recursos para realizar los desarrollos sería lo ideal, pero en este proyecto el objetivo principal es realizar el símil de cómo un niño podría ser capaz de realizar un proceso de reconocimiento de palabras, fonemas o frases prácticamente desde cero.

1.2 Procesamiento con recursos limitados

El procesamiento del habla con recursos limitados se centra en el proceso de decodificación con poca o nula información lingüística. Como se ha presentado anteriormente se han hablado de tres cosas: Un modelo acústico asociado a cada fonema o palabra del lenguaje, el diccionario de palabras que lo conforman con sus respectivas asociaciones y un modelo del lenguaje con el que estamos trabajando con probabilidades y secuencias.

Se cuentan con dos estrategias moderadamente diferentes dependiendo de la cantidad de recursos de que disponemos: Con ningún recurso (*Zero resources*) o con algo de información (*Low resources*).

- *Zero resources*. Como su nombre indica, no tendremos a nuestra disposición ningún recurso lingüístico previo para poder realizar un entrenamiento a nuestro sistema con métodos de aprendizaje automático, por ejemplo no disponer de la transcripción del audio de entrenamiento. Se parte de cero y lo único de que se dispondrá es de las características acústicas del audio de entrada. En este punto se situaría el escenario principal de nuestro proyecto.

- *Low resources.* En este tipo de escenario no se parte desde cero, se tendrá una cierta cantidad de recursos parciales, para su mejor entendimiento en el caso particular de este proyecto en nuestro escenario secundario se tendrá posesión de una determinada cantidad de palabras que serán comparadas con un audio principal con el fin de ser encontradas en él. También se pueden utilizar recursos lingüísticos "universales", de lenguas distintas a que se esté trabajando.

La primera parte del proyecto estará basada parcialmente, como se acaba de comentar, en la modalidad *Low resources*, mientras que el apartado principal sigue una estrategia de *Zero resources*.

1.3 Objetivos del proyecto

El objetivo principal de este proyecto es el desarrollo de un sistema de procesamiento de lenguaje natural el cual permita la extracción de la información de un discurso hablado. Dicho discurso habrá sido previamente procesado pero no se dispondrá de ningún recurso propio del idioma a tratar, simulando así el proceso que un niño pueda realizar al dar los primeros pasos para aprender un idioma.

Con la finalidad de alcanzar dicho objetivo, el proyecto será dividido en varias partes:

- Desarrollo de un sistema que a partir de un conjunto de palabras sea capaz de localizarlas en un audio extenso. (*Query by example*)

- Desarrollo de un sistema que comparando dos audios, sea capaz de extraer e indicar la posición en dichos audios de palabras similares. (*Zero resources spoken term discovery*)

1.4 Estructura de la memoria

La memoria ha sido dividida en cinco capítulos: Introducción, estado del arte, diseño de la solución al primer problema, diseño de la solución al segundo problema y un último apartado con las conclusiones extraídas del proyecto.

Tras esta introducción a los diferentes temas a tratar, se abordará el estado del arte, donde además de exponer los principales algoritmos para la resolución de este tipo de

problemas, se realizará una crítica que justifica el desarrollo del sistema que se propone.

El tercer capítulo trata dentro del escenario de *Query by example* introducido anteriormente, el primer problema que se debe abordar. Presentará los pasos seguidos en el diseño del sistema, los problemas aparecidos durante el desarrollo y las medidas tomadas para su resolución. Por último se dará a conocer el algoritmo final y sus resultados con pruebas reales.

El tercer capítulo ampliará el problema presentado anteriormente, siendo introducidos en el escenario de *Zero resources spoken term discovery*, lo que hará que la solución anterior sea inviable. Se realizará una introducción a unos nuevos conceptos que se incluirán en este apartado, explicando posteriormente los pasos seguidos durante el desarrollo y la forma de resolver los problemas aparecidos. Por último serán presentadas las pruebas y la evaluación del desarrollo.

El escrito finaliza exponiendo las conclusiones a las que se ha llegado con el desarrollo, adicionando una pequeña vista hacia el futuro en el campo del procesamiento del lenguaje con recursos limitados y de qué forma se podría dar continuidad a este proyecto.

Capítulo 2
Estado del arte

Los dos tipos de algoritmos principales que se encuentran dentro del campo del *machine learning* o aprendizaje automático son: supervisados y no supervisados.

- Algoritmos supervisados. Para este tipo de algoritmos, se les proporciona una entrada de información etiquetada de forma correcta a modo de ejemplo, de forma que el sistema diseñado "aprende" de dicha información y es entrenado. Una vez finalizado el entrenamiento el sistema es capaz a través de la información que ha archivado, de relacionar las nuevas entradas que le sean proporcionadas y realizar el proceso para el que ha sido diseñado.

- Algoritmos no supervisados. En este tipo, al sistema creado no se le facilita ningún tipo de información etiquetada correctamente para que la procese y la tome como referencia, si no que se le presenta el problema y ha de resolverlo con la información de que dispone en ese momento preciso.

En el procesamiento de audios con recursos limitados, los algoritmos a utilizar son los no supervisados. Esto es causado porque los escenarios presentados no se dispone de la información suficiente para construir un modelo de datos completo, bien porque la obtención de esa información conlleva un gasto inasumible, porque se trate de un idioma del que no se tiene suficiente información o porque se está tratando de simular el aprendizaje desde cero de una persona con un idioma desconocido o un niño.

La base que se ha establecido en los últimos años y que aún es respetada en las implementaciones actuales comienza con el algoritmo no supervisado presentado en *(Park y Glass, 2005)*. Dicho algoritmo permitía descubrir patrones acústicos a partir de repeticiones en diferentes audios. Dicho algoritmo es una adaptación del algoritmo de programación dinámica conocido como *Dynamic Time Warping* (DTW).

Presentando dos secuencias de audio preprocesado a dicho algoritmo, nos es retornada la alineación óptima entre ellas. A continuación podemos observar la fórmula recursiva de dicha ecuación:

$$M(i,j) = \begin{cases} 0 & i = j = 0 \\ +\infty & i = 0,\ j > 0 \\ +\infty & j = 0,\ i > 0 \\ \min_{\forall (x,y) \in S} M(i-x,\ j-y) + D(A_i,\ B_i) & i > 0 \end{cases}$$

Algoritmo 1. Park & Glass, 2015

- M corresponde con la matriz de programación dinámica.

- S es el conjunto de movimientos permitidos, representados como pares de incrementos horizontales y verticales (x, y).
- $A_i B_i$ son los objetos que representan a las posiciones i-ésima y j-ésima de sus respectivas secuencias.

- D es la función que calcula la distancia entre los dos objetos.

El coste temporal y espacial del algoritmo es proporcional a la longitud de los dos audios que pretendemos comparar, por lo tanto es $\Theta(nm)$, siendo n y m la longitud de los audios.

El problema de dicho algoritmo, el cual también presentará nuestro primer modelo desarrollado, se produce cuando el audio está compuesto por múltiples palabras componiendo una frase, lo cual no permite aplicar el DTW tal y como fué presentado anteriormente ya que la comparación del algoritmo era de un audio contra otro situando el inicio y final de ambos audios en los mismos puntos del tiempo, por lo tanto si la palabra a encontrar no está en el mismo espacio temporal no será detectada.

La variación propuesta por *Park y Glass* es permitir al algoritmo recoger alineamientos locales o subsecuencias, lo cual hace que puedan existir diferentes caminos óptimos en diferentes subsecuencias de la matriz principal. Su funcionamiento es el siguiente:

1. Se define una variable W, la cual hace referencia a la anchura de las bandas diagonales de nuestra matriz de distancias.
2. Se aplica el algoritmo DTW sobre cada una de esas bandas, para encontrar el mayor alineamiento local.
3. Se define un límite superior para el alineamiento, de forma que un alineamiento local no será válido si está por encima de un determinado valor umbral L.
4. Por último, analizamos los resultados y nos quedaremos con el alineamiento correspondiente a la menor distancia local.

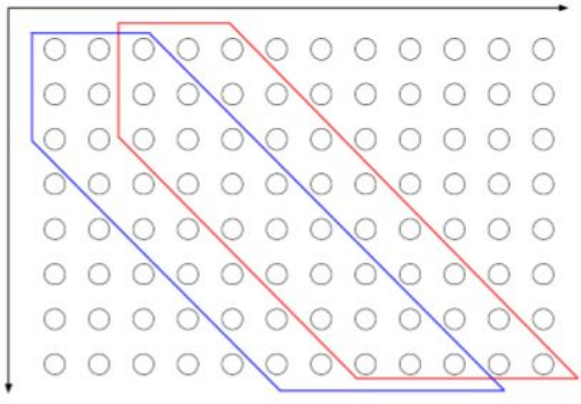

Figura 2. Park y Glass Segmental DTW (W = 2)

Problemas que presenta esta solución:

- El coste computacional es mucho mayor al coste del DTW inicial. Para encontrar la subsecuencia de menor distancia con la restricción de que esta subsecuencia debe tener una longitud mínima $(Lmin)$ y una longitud máxima $(Lmax)$. el coste es $\Theta(Lmax * m)$. Por lo tanto el coste de la obtención y análisis de cada una de las subsecuencias multiplicaría al coste anteriormente mencionado dependiendo del tamaño de las mismas $\Theta(n^2 * Lmax * m)$.

- Dentro del desarrollo de esta segunda parte, hay dos parámetros importantes que definen la precisión del algoritmo $(W\ y\ L)$ los cuales son definidos a priori y pueden no contar con la precisión adecuada.

A pesar de estos dos puntos, la gran mayoría de los sistemas posteriores fueron basados en este algoritmo.

(*Muscariello et ál, 2012*) presentaron más tarde una modificación del algoritmo visto anteriormente llamado *Segmental Locally Normalized DTW*. La distinción se produce al valorar la distancia del punto de inicio del DTW y la longitud de los caminos durante la minimización. Podríamos dividirlo en tres pasos:

1. Identificación de los posibles puntos de inicio donde puedan comenzar los patrones similares a partir de la distancia inicial entre ellos.
2. Calcular el alineamiento de los caminos que comienzan en los puntos anteriores teniendo en cuenta la distancia normalizada por la longitud del camino.
3. Asignar una puntuación como medida de la similitud de los dos fragmentos correspondientes al alineamiento obtenido.

Esta aproximación es la base que se ha tomado como referencia para la segunda parte del proyecto y se realizará una explicación más extensa en dicho apartado.

Por último, (*Jansen et ál, 2011*), buscando la eficiencia tanto en memoria como en la búsqueda de los patrones, introducen tres nuevos algoritmos que conjuntamente son capaces de obtener una matriz de similitud con coste $\Theta(n\ log\ n)$. Dichos algoritmos son:

- *Locality Sensitive Hashing.* En esta primera fase se realiza un preproceso de la representación vectorial de los audios, de forma que se pase de un espacio R^d a otro $\{0,1\}^b$.
- *Point Location in Equal Balls.* Esta fase es la que ocupa el coste principal del algoritmo, ya que es donde se realiza el cálculo de la matriz de similitud aproximada a partir de los vectores obtenidos en el proceso anterior.
- *Two-Pass Repeated Trajectory Search.* Por último para buscar los patrones en la matriz que nos devuelve el proceso anterior, se usa un algoritmo de dos pasos que busca líneas diagonales y a las que les aplica un simple DTW.

Como podemos observar al proceso principal de DTW, con el fin de optimizar su funcionamiento, le han sido añadidos generalmente preprocesos y postprocesos de forma que la información ya haya sido tratada previo a la realización del algoritmo principal con el se obtiene el resultado.

2.1 Crítica al estado del arte y propuesta

Las aportaciones mencionadas en los apartados de la sección anterior muestran grandes avances realizados en el campo del procesamiento de audios y del procesamiento del lenguaje natural.

La base de los dos algoritmos que va a ser desarrollada y adaptada según las necesidades del proyecto es la realizada por *(Park y Glass, 2005)* y *(Muscariello et ál, 2012).*

Y tal y como ellos presentan en sus escritos, serán encontrados diferentes problemas a resolver, sobre todo en el campo de la eficiencia algorítmica. El coste computacional de los algoritmos base presentado por ellos es relativamente alto y en las computadoras normales el tiempo real de proceso de dichos algoritmos con unos audios de longitud considerable sería muy alto.

Capítulo 3
Query by example

3.1 Query by example

En este capítulo se va a realizar la presentación del primero de los problemas que aborda este trabajo. La tarea es conocida como *Query By Example*.

Primero va a ser presentado el problema y el objetivo principal que se espera conseguir, posteriormente serán definidos los diseños de las soluciones y la forma de llegar a ellos, después serán presentados los resultados del algoritmo y por último serán expuestas las conclusiones a las que se ha llegado durante el desarrollo.

La base en la que el sistema va se va a apoyar es la ya mencionada en el estado del arte, *(Park y Glass, 2005)*. A modo de recordatorio, dicho algoritmo tenía la particularidad de permitir recoger alineamientos locales o subsecuencias dentro de la matriz de distancias principal, lo cual hace que puedan existir diferentes caminos óptimos o en otras palabras nos permite detectar dentro de un mismo audio extenso una misma palabra en diferentes espacios de tiempo.

La segunda variable del DTW que va a ser introducida es la diseñada en *(Müller, 2007)* en el libro *Information Retrieval for Music and Motion*. En dicho libro describe cómo el algoritmo base del que hablamos en los puntos anteriores, DTW o *Dynamic Time Warping*, pretende encontrar el mejor alineamiento de una secuencia dentro de otra secuencia de longitud mayor, lo cual es exactamente lo pretendido en esta tarea. Por hacer el símil real, nuestra subsecuencia es una palabra y la secuencia de longitud mayor un discurso. La modificación de DTW presentada por Müller trataría de encontrar dentro de ese discurso las veces que es repetida nuestra palabra.

Müller denomina a esta variante Subsequence DTW. Su distinción principal es que el alineamiento puede comenzar en cualquier punto de la subsecuencia de mayor longitud.

Apoyado por ambos algoritmos mencionados anteriormente, vamos a proceder a exponer de forma completa la tarea a realizar, los objetivos a conseguir, el material del que disponemos para trabajar, el enfoque dado al problema junto con el diseño de la solución y por último las conclusiones a las que se han llegado durante el desarrollo y tras el análisis de los resultados.

3.2 Descripción de la tarea

El objetivo principal de la tarea es buscar en un conjunto de audios de mayor longitud, una determinada serie de palabras definidas en audios más cortos que los anteriores. El sistema a desarrollar ha de ser independiente del idioma.

En nuestro caso particular, el material del que disponemos es el siguiente:

- Cien subsegmentos de audio que corresponde cada uno con una palabra. Han sido previamente parametrizados. Cada subsegmento es un fichero de texto plano compuesto por vectores de 186 elementos.
  Ésta representación de las características acústicas ha sido realizada considerando un conjunto de unidades fonéticas que no tienen nada que ver con el lenguaje utilizado. En concreto cada vector representa la probabilidad de pertenencia del frame a cada una de las 186 unidades consideradas.

- En cuanto a la relación vectores/segundo en los audios, se muestrea cada 10ms dando un total de 100 vectores por segundo.

- Dos audios largos que corresponden a dos programas de televisión.

Con el fin de facilitar la evaluación de los resultados obtenidos, los audios y las transcripciones de las palabras son en castellano, pero el desarrollo debería ser aplicable a cualquier idioma independientemente de si es reconocido por el desarrollador o no.

3.3 Enfoque del problema y diseño de la solución

Como se ha mencionado durante el inicio del documento, la mayoría de sistemas están basados en el algoritmo DTW, el cual es va a ser la base también para el funcionamiento de nuestro sistema.

En concreto el modelo utilizado podría considerarse un híbrido entre el algoritmo de Müller y el de Park y Glass.

El punto diferencial principal de los algoritmos anteriores era la introducción de un nuevo parámetro W el cual hacía referencia a la anchura de la banda diagonal de la matriz de distancias.

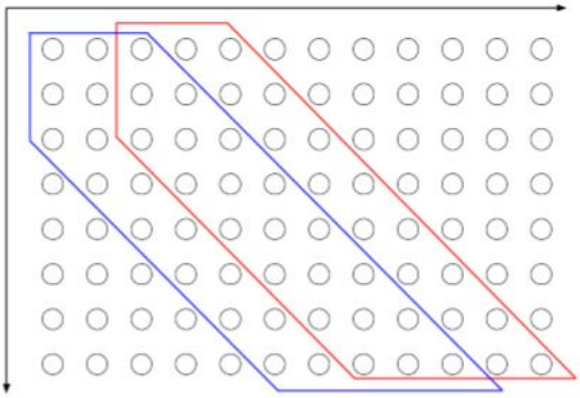

Figura 3. Segmental DTW (W = 2)

Usando ese nuevo concepto, se va a llevar al extremo en el desarrollo, escogiendo como valor para la W la amplitud total del audio más corto (palabra) para crear así subsegmentos dentro del audio principal para realizar las comparaciones, realizando la comparación de la palabra en cada uno de los puntos temporales del audio principal.

Explicándolo en un ejemplo paso a paso. Tamaño adaptado a la explicación.
Discurso largo de tamaño: 12
Palabra de tamaño: 3

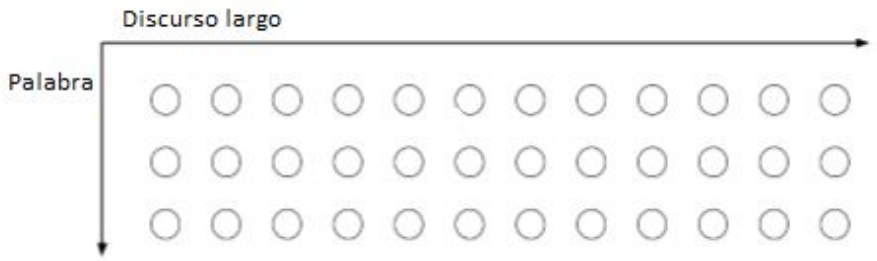

Figura 4. Inicio matriz de distancias (Query By Example)

1. El primer paso es definir el tamaño de W, el cual está definido como el tamaño completo de la palabra a comparar $(W = 3)$.
2. Las comparaciones comenzarán desde el inicio del discurso largo, quedando la primera matriz como muestra la Figura 5.

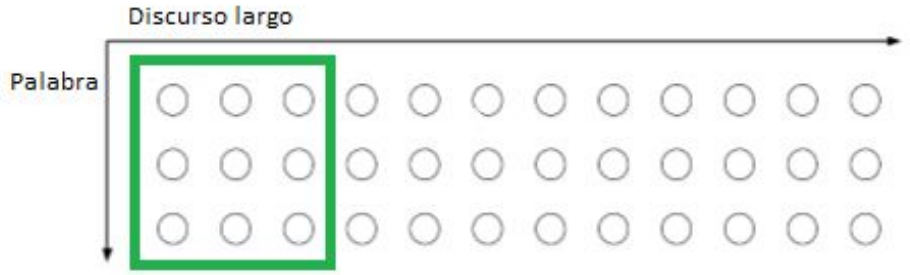

Figura 5. Segundo paso matriz de distancias (Query By Example)

3. Obviando el resto de la matriz, se realiza el algoritmo DTW sobre el cuadrado marcado en verde, buscando el camino óptimo dentro de dicha submatriz.
4. Será definido un umbral, de forma que solamente serán guardados como resultados los caminos óptimos donde la distancia sea inferior a dicho umbral. Si el resultado obtenido está por debajo del umbral, ese segmento recogido del audio principal corresponde con la misma palabra de la que estamos haciendo la comparación.
5. Por último una vez terminado el proceso, se mueve la matriz un valor en el tiempo, un campo a la derecha en nuestro ejemplo.

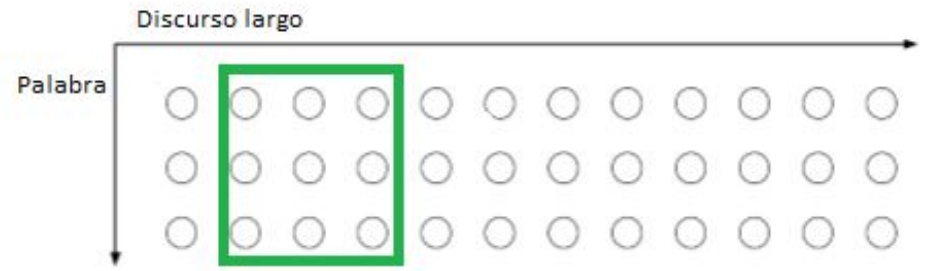

Figura 6. Quinto paso matriz de distancias (Query By Example)

6. Y se repite el proceso desde el punto 3 hasta que se finaliza el recorrido del discurso largo.

Con este algoritmo, se está recorriendo el audio más largo realizando un DTW con la palabra a comparar por cada uno de los frames de tiempo que dura el discurso. Por lo que estamos encontrando si la palabra aparece en cualquier momento del discurso y cuantas veces aparece, teniendo varios resultados dependiendo de la precisión que queramos darle a nuestro desarrollo.

Lo más probable es que las palabras en ambos discursos no tengan la misma duración y con el funcionamiento del algoritmo explicado se está forzando a que sea así, sin darles un margen estricto de longitud. Para ello se hace un postprocesamiento una vez que se tienen calculados todas las distancias y todos los candidatos, de forma que se evalúa el entorno del candidato para comprobar si a su alrededor tiene puntos de baja distancia o no antes de seleccionarlo.

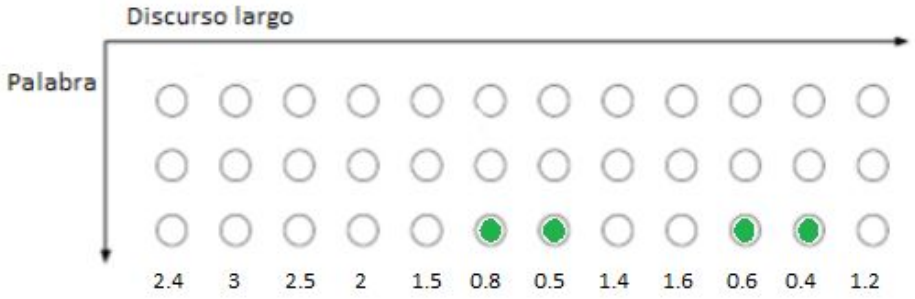

Figura 7. Postprocesamiento 1 (Query By Example)

En la Figura 7 se puede observar como los valores de las distancias se aproximan a 0 bajando su distancia cuando hay un candidato alrededor. También podemos visualizarlo en el siguiente gráfico (Figura 8) que representa el vector de resultados:

2,40	3,00	2,50	2,00	1,5	0,80	0,50	1,40	1,60	0,60	0,40	1,20

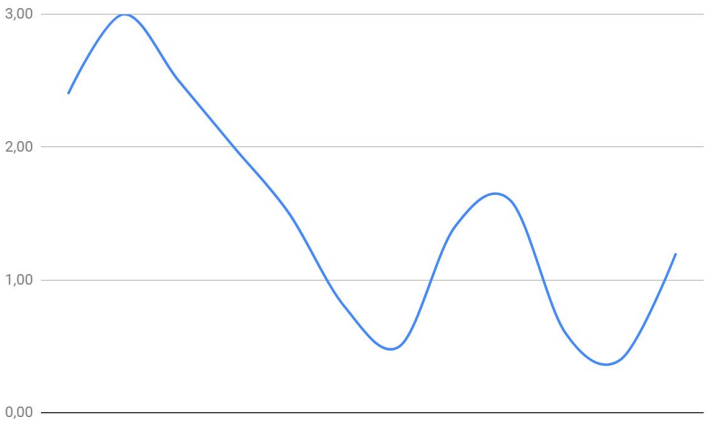

Figura 8. Postprocesamiento 2 (Query By Example)

En la Figura 8, se aprecia de forma visual cómo habrían dos resultados correctos dada las dos depresiones marcadas en el gráfico.

3.4 Experimentación y resultados

3.4.1 Caso 1

Para la experimentación y pruebas del algoritmo desarrollado iniciaremos la primera prueba con uno de los subsegmentos que corresponde a una palabra.

- Discurso: Un audio de programa de televisión previamente preprocesado y convertido a vectores. Ocupa 31 MB y dura aproximadamente 3 horas.

- El subsegmento a utilizar para hacer la comparación en primera instancia va a ser “SEG-0001 amigos”, donde la palabra a comparar como indica el nombre del fichero es “Amigos”.

Ambos son, como mencionamos al inicio del capítulo, ficheros de texto plano compuestos por vectores de 186 elementos. A continuación podemos ver una imagen de los 5 primeros elementos de los 5 primeros vectores.

```
1  7.220235e-04 2.422489e-03 2.343352e-01 2.852869e-04 3.805703e-06
2  3.392880e-03 2.970003e-03 3.452788e-01 6.564053e-05 5.980330e-07
3  9.486479e-04 5.189832e-04 4.361229e-02 1.699972e-05 4.653102e-07
4  1.483173e-03 5.825975e-04 4.329634e-03 1.045368e-05 4.079413e-07
5  3.131952e-05 2.993722e-06 6.616771e-05 5.736835e-06 9.732761e-08
```

Figura 9. Ejemplo vectores (Query By Example)

En total el discurso está formado por 13.005 vectores y el subsegmento que hace referencia a la palabra ocupa 45 vectores.

El umbral aplicado al algoritmo en esta prueba se calcula de forma experimental. Incrementando este valor aparecen muchos más casos con mayor probabilidad de ser incorrectos pero si reducimos demasiado dicho valor perdemos gran cantidad de soluciones.

En esta primera prueba obtenemos 48 resultados. Hay que tener en cuenta que estamos situados en el final de la matriz por lo que el punto *X* siempre será 44 (El tamaño del subsegmento) y el punto *Y* hará referencia al lugar de la matriz de distancias donde el camino es el más óptimo (punto final, donde la palabra ya ha terminado). A modo de ejemplo sencillo la siguiente imagen:

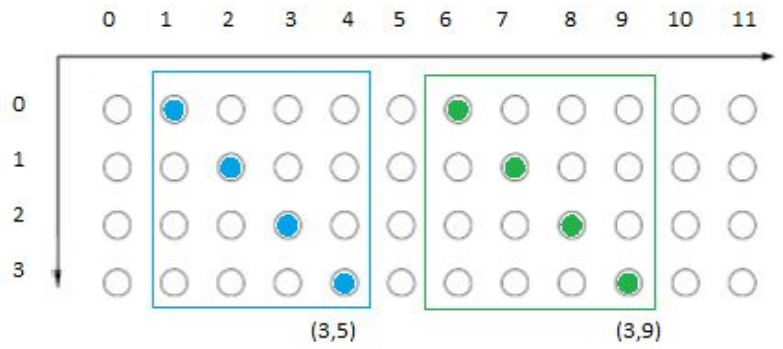

Figura 10. Caso 1 (Query By Example)

Al finalizar el algoritmo principal, se obtienen 48 resultados correctos, pero como ya hemos comentado antes, las palabras pueden alargarse o acortarse en el tiempo, por lo que hay que aplicar el postprocesamiento mencionado anteriormente. Solamente se han valorado como resultado final, los casos donde además de tener un positivo, a su alrededor también había uno o más puntos con una distancia inferior al umbral.

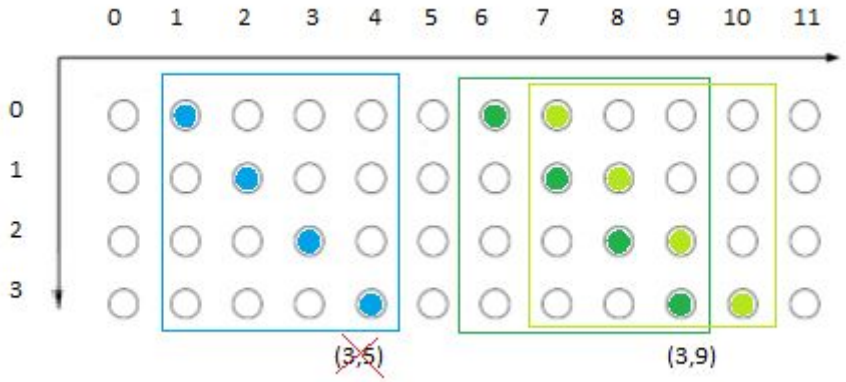

Figura 11. Caso 1 postprocesamiento (Query By Example)

Los resultados finales de esta prueba han sido:

(44, 1980), (44, 1981), (44, 1984),
(44, 4259), (44, 4260), (44, 4261), (44, 4262),
(44, 4467), (44, 4468), (44, 4469), (44, 4470),
(44, 5437), (44, 5438), (44, 5439),
(44, 7357), (44, 7358), (44, 7359),
(44, 7596), (44, 7597),
(44, 8975), (44, 8976),
(44, 9516), (44, 9517), (44, 9518),
(44, 10682),(44, 10683),(44, 10684),
(44, 10873),(44, 10874),
(44, 11229),(44, 11230),(44, 11231),
(44, 11753),(44, 11854),
(44, 12228),(44, 12229),

3.4.2 Caso 2

El testeo anterior representa el caso más común, donde sí se encuentran valores por debajo del umbral. La palabra que se había utilizado en el ejemplo anterior era “amigo”, la cual es bastante común.

Dentro de las 100 palabras que se han buscado dentro del audio principal, hay ciertas de las que no hemos encontrado ninguna coincidencia. En ese caso nuestro algoritmo nos devuelve simplemente una tupla vacía, ya que no se ha dado ningún caso durante el proceso en el que la distancia total fuese inferior al umbral.

Es el caso de los segmentos:

- 3: Andalucía.
- 4: Bruselas.
- 6: Capacidad.
- 9. Claramente.
- 18. Democracia.
- ...

Ambos tipos de respuestas son consideradas válidas, ya que probablemente la palabra a comparar es cierto que no fué utilizada.

3.5 Conclusiones

Para esta prueba se ha tratado de seguir, adicionando unas pequeñas variantes, el algoritmo de *Park y Glass* junto con *Müller*. Los resultados parecen correctos y con sentido, pero sin un método completo de evaluación no podemos conocer el porcentaje total de aciertos y de errores en la solución.

Capítulo 4
Zero resources spoken term discovery

4.1 Diseño de la solución final

Tras el desarrollo del punto anterior y los resultados extraídos del mismo, se ha concluido mencionando los problemas de falta de eficienci a así como la carencia de una evaluación más completa con el fin de verificar los resultados obtenidos. Se va proponer un nuevo caso en el cual la solución anterior sería tan costosa algorítmicamente que no se puede obtener un resultado viable, visible ni evaluable.

Dicho problema y la solución a aplicar, es una variación del algoritmo de DTW para entornos de *zero resources* presentado en *(Muscariello et ál, 2012)*.

En el apartado anterior el objetivo principal era a partir de una serie de subsegmentos que hacían referencia a palabras, realizar la comparación de estas palabras con un discurso largo con el fin de encontrar las veces que se repite cada una de las palabras junto con sus posiciones dentro del discurso *(Query by example)*.

En este segundo problema, no se tendrá disponible el diccionario de palabras para realizar las comparaciones. A partir de dos audios completos, en este trabajo se han usado frases de aproximadamente 10 segundos, vamos a realizar una comparación, audio contra audio con el fin de encontrar en dichos discursos palabras que se repitan en ellos.

El problema principal que se tratará de abordar es la eficiencia de los procesos para que el proceso, sin perder precisión de resultados, pueda llegar a ser mucho más rápido.

4.2 Descripción de la tarea

El objetivo principal de la tarea es a partir de dos segmentos completos de audio en los cuales se dice una frase, realizar la comparación entre ambas frases y extraer en que posiciones de ambos audios coincide que se repita una misma palabra o varias. Es decir:

Frase1: Quiero hablar con el jefe de granada por favor.
Frase2: Desearía hablar con el uno nueve siete cinco.

De estas dos frases, al realizar la comparación, se debería retornar como resultado la palabra “hablar” y la posición en la que está situada en dichos discursos.

El material del que se dispone es el siguiente:

- Doscientos audios que contienen diferentes frases cortas. Dichos audios igual que en el problema anterior, han sido previamente parametrizados y descompuestos en ficheros de texto con “X” vectores de 39 elementos, cada vector compuesto a su vez por trece vectores de coeficientes cepstrales con su primera y segunda derivada.

- La transcripción de cada uno de los audios con las frases exactas que se dice en ellos. Solamente serán utilizados en la evaluación.

- Los ficheros .wav para escuchar los audios. Solamente utilizados en la evaluación como información adicional.

Igual que en el caso anterior, para una mayor facilidad durante el desarrollo y para realizar las pruebas y evaluación, se han utilizado frases cortas usando el castellano como lenguaje. Una vez finalizado el algoritmo podría ser extrapolado a cualquier idioma.

4.4 Enfoque del problema y diseño de la solución

4.4.1 Introducción a un nuevo concepto. *Thresholding*

Uno de los problemas que tiene la comparación entre vectores de características acústicas es la poca discriminación en determinadas zonas de audio. Además, si se consideran todas las posibles distancias entre frames de ambos audios la cantidad de información a procesar es extremadamente grande creando un problema de eficiencia computacional. Dada la gran cantidad de información a analizar y el nuevo objetivo de esta segunda parte, vamos a introducir un nuevo concepto llamado “*Thresholding*” o “Umbral”.

El *thresholding* es comúnmente utilizado en los algoritmos de visión por computador con el fin de reducir al máximo la información previa a la aplicación del algoritmo principal. Es el principal método de segmentación de imágenes. Un algoritmo de *thresholding* reemplaza cada uno de los pixeles de una imagen con un pixel de intensidad máxima (negro) si supera cierto umbral predefinido, o un píxel de intensidad minima (blanco) si no supera dicho umbral.

Siendo el umbral de *thresholding T* la definición de su algoritmo es:

$$g(x,\ y) =\ 1\ if\ (x,y)\ > T\, g(x,\ y) =\ 0\ if\ (x,y) \leq T$$

Algoritmo 2. Thresholding

A continuación va a ser explicado su funcionamiento más común en un sencillo ejemplo de visión por computadora.

La Figura 12 corresponde con la imagen inicial que va a ser procesada, representa una imagen microscópica de células blancas.

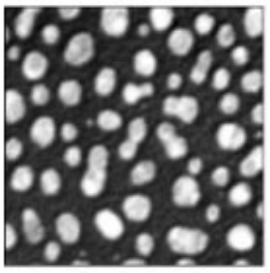

Figura 12. Thresholding 1

La Figura 13, contiene la distribución de la intensidad de los pixeles, comprendida entre 0 y 255. Esta distribución es muy sencilla y se puede apreciar perfectamente que si el valor de *thresholding* se sitúa en la mitad *(T=127)* podremos dividir nuestra información en dos grupos bien diferenciados.

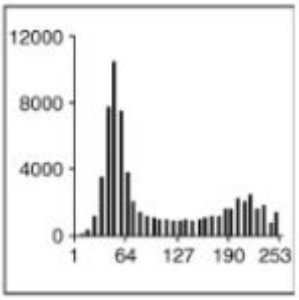

Figura 13. Thresholding 2

Una vez hecha la división, se formatea la imagen, y los puntos por encima del valor *T* se representarán con la intensidad máxima (negro) y los de menor *T* con la mínima (blanco), apareciendo la Figura 14.

Figura 14. Thresholding 3

Este algoritmo como se ha mencionado, es uno de los procesos más comunes en visión por computadora. Para poder aplicar este algoritmo en el proyecto es necesario disponer de la representación de los audios en una imagen, lo cual nos lleva a los siguientes conceptos.

4.4.2 Proceso de DTW típico

Para mejorar la comprensión de las modificaciones al algoritmo de DTW genérico que van a ser realizadas, se va a explicar al detalle el funcionamiento del proceso de DTW.

Para poder situar espacialmente en una imagen el algoritmo de *Dynamic time warping* se colocan cada uno de los audios a comparar a un lado de la matriz. Partiendo desde el primer punto de dicha matriz se va a realizar un recorrido por ella navegando por el camino el cual la distancia vectorial entre los puntos de un audio y el otro sea menor.

En la imagen inferior se ha representado el recorrido de un algoritmo de *DTW* típico detenido a mitad del proceso.

- Cada uno de los ejes, *X* e *Y* representa un audio desplegado en el tiempo.
- Los números que tienen cada uno de los puntos ponen puntuación a la distancia entre el segmento del audio 1 con el audio 2. Dicha puntuación se extrae a partir de comparar el punto en el que estamos situados con los de su alrededor.

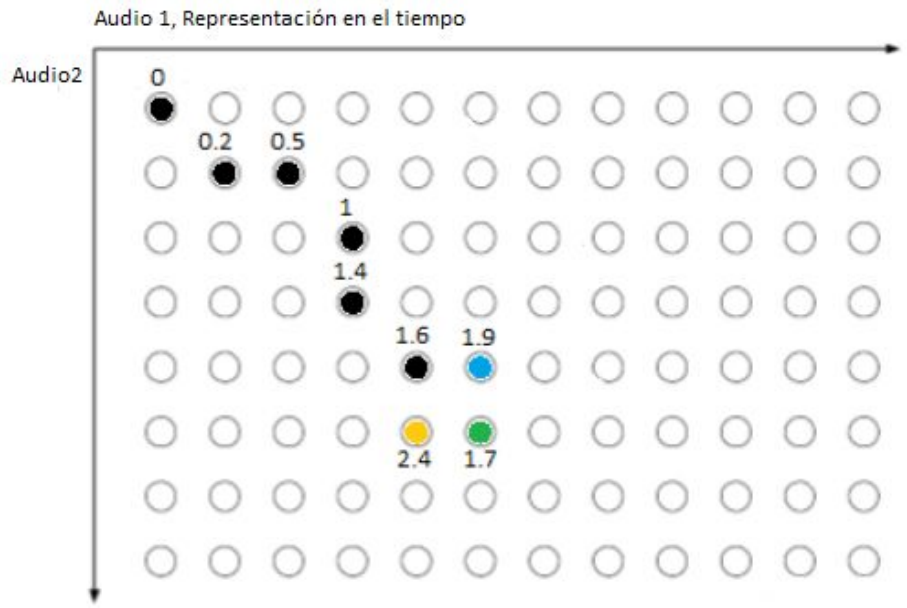

Figura 15. Proceso DTW

En la Figura 16 el puntero del proceso se encuentra en el punto de valor 1.6, donde se puede observar que ha sido extraída la distancia a los puntos de su alrededor a partir del valor propio del punto y de su entorno. De esta forma se puede saber claramente cuál debería ser el próximo destino en el camino, el punto verde de valor 1.7.

El siguiente paso en el proceso sería situarnos en dicho punto de 1.7, volver a calcular a su alrededor el resto de puntos, escoger el de menor valor y continuar con el algoritmo.

Conociendo estos los dos puntos anteriores ahora se va a proceder al análisis de la tarea y de cómo van a ser aplicados en conjunto para su resolución.

4.4.3 Diseño de la solución

El objetivo del desarrollo es el realizar la comparación entre dos audios, y detectar palabras iguales en dichos audios. Ya se han presentado los conceptos de *thresholding* y se ha hecho una breve explicación del funcionamiento del algoritmo de DTW.

En el punto inicial de la solución, se parte de dos audios ya preprocesados y divididos en vectores, los cuales para compararlos los situaremos a modo de matriz exactamente igual que en el inicio del algoritmo de *DTW*. Dicha matriz va a ser el escenario con el que se va a empezar a trabajar previo a la introducción del proceso principal de DTW.

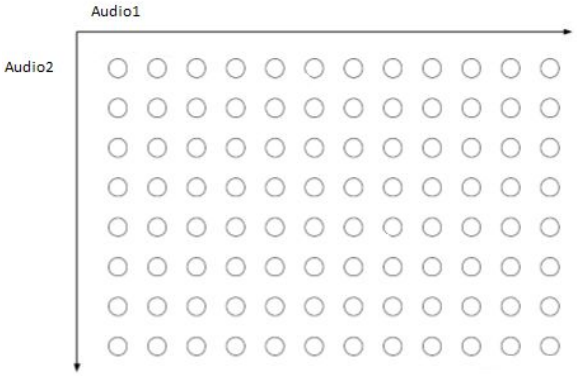

Figura 16. Matriz de distancias

La problemática que presenta el problema es que a diferencia del *Query by example*, en este caso cualquier punto de la matriz puede ser un punto inicial del algoritmo *DTW* ya que al estar compuesto por varias palabras cada audio, hay que valorar la idea de que alguna palabra en un audio se diga al final y en otro al principio. Por lo tanto si se tuviera que utilizar el proceso de la tarea anterior primero habría que:

- Seleccionar el tamaño aproximado que podría componer una palabra para usarlo de submatriz para la comparación. También se podría seleccionar dicho recuadro de forma dinámica pero se estaría adicionando un nuevo proceso con su respectivo tiempo de procesamiento.

- Anteriormente solamente se recorría en un eje dicho recuadro. Aquí entra en juego una nueva dirección por lo que el coste del algoritmo se vería multiplicado.

- Tras el algoritmo principal, habría que postprocesar la información utilizando aparte del proceso que se realizaba al final el *Query by example,*nuevos filtros para, en este caso, verificar los puntos de alrededor en todas las direcciones espaciales, no solo en horizontal.

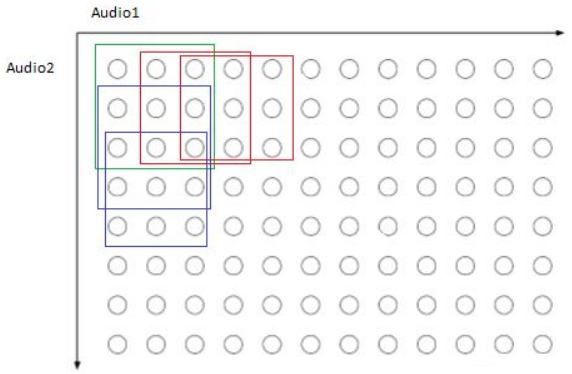

Figura 17. Proceso Query by example en Matriz Distancias

Por todo lo anterior el proceso es muy costoso. A continuación se introducirán los nuevos conceptos explicados en este capítulo modificando el comportamiento del proceso.

Lo primero que realiza nuestro algoritmo es un preprocesamiento de la información. Van a ser comparados los dos audios el uno contra el otro sin aplicar aún ningún algoritmo, solamente se calcula la distancia vectorial punto a punto y por separado, de forma que antes de empezar el algoritmo tendremos disponible una matriz de distancias completa. En este filtro se está perdiendo precisión en los resultados al no tener en cuenta el entorno del punto en el que se está situado para conocer la distancia al resto de ellos, pero a cambio se ganará eficiencia al no estar realizando el cálculo de la distancia de forma dinámica.

La representación en una imagen de la matriz de distancias de un fichero real sería la mostrada en la Figura 18:

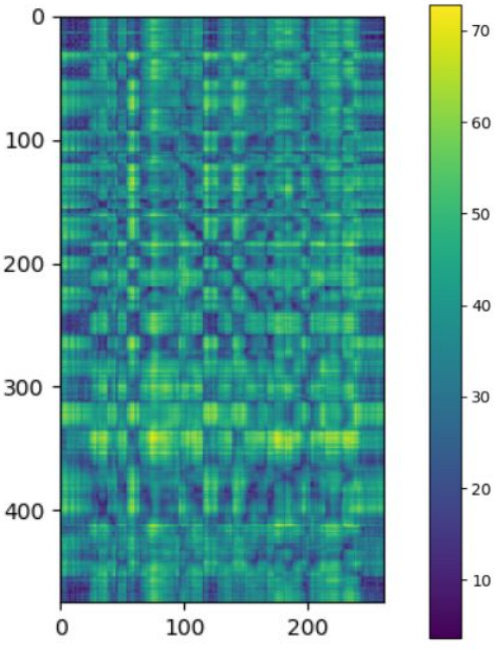

Figura 18. Matriz de distancias ejemplo (Representación)

Los puntos más oscuros son donde la distancia vectorial es más baja, es decir si el punto X del audio 1 y el punto Y del audio 2, punto *(X,Y)* de la matriz tiene un color oscuro, significa que en ese momento los dos audios tienen vector igual o parecido. Si se juntan muchos de estos puntos consecutivos, se podría encontrar un fonema parecido e incluso una palabra parecida.

El segundo paso del proceso es aplicar a la matriz de distancias el concepto de *thresholding*. Es decir se va a definir un umbral *T* que representará la precisión que queremos darle a nuestro algoritmo. En este desarrollo en particular los valores por debajo del umbral *T* serán los que se deben mantener representados en la imagen, y los que están por encima serán coloreados en blanco, eliminando así el "ruido" de la matriz.

Como adición al proceso de *thresholding* y con el fin de mejorar la precisión, se han definido dos umbrales, uno muy bajo para encontrar los puntos donde ambos audios coinciden casi a la perfección y un segundo filtro más laxo donde se permite una mayor diferencia, de forma que si se producen pequeñas variaciones en la pronunciación de la palabra, o si esta misma está alargada en el tiempo o dicha más rápidamente, se pueda también detectar. Aplicando este filtro a la representación gráfica de la matriz de distancias anterior obtenemos la Figura 19:

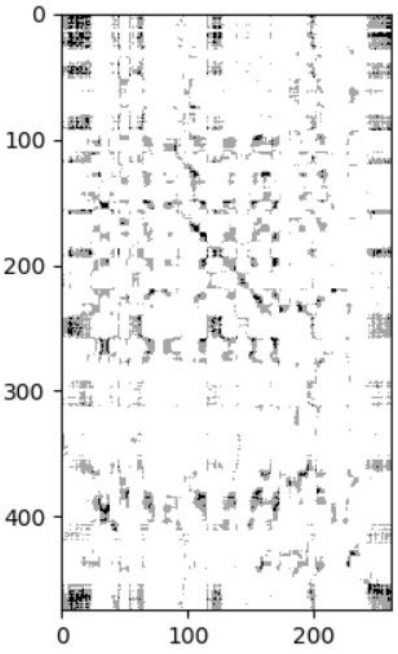

Figura 19. Matriz de distancias ejemplo (Thresholding)

Como se puede apreciar, la información contenida es considerablemente menor y visualmente se puede llegar a reconocer subsegmentos donde pudiera existir una palabra, pero sin poder confirmarlo.

En el próximo paso se aplicará el algoritmo principal de *DTW*, pero con una modificación, ya que los caminos a recorrer ya los tenemos marcados y acotados.

Los puntos negros de la Figura 20 son los puntos por debajo del umbral *T* principal, es decir donde se está prácticamente seguro de que en ambos audios dicho punto es igual. Se va recorrer la matriz por completo, pero solamente se lanzará el *DTW* en dichos puntos.

Cuando sea encontrado uno de esos puntos, el *DTW* a aplicar es más sencillo que el utilizado previamente ya que aparte de tener los puntos negros de umbral bajo, también tenemos los grises, los cuales representan un segundo umbral, el cual no es tan bajo como para ser un caso exacto pero es lo suficientemente preciso como para tenerlo en cuenta.

Por lo tanto los puntos negros serán el inicio del recorrido, y dicho recorrido será correcto siempre y cuando el siguiente punto a del camino sea reconocido como negro o gris. Si en algún momento el siguiente punto del camino es blanco, se anula dicho punto y se cancela en ese punto el *DTW*.

La tercera condición para reconocer el matching completo de una palabra es el tamaño del camino encontrado, esto se define manualmente y en cada una de las pruebas será incluido el valor tomado. Particularmente en este ejemplo, un camino será reconocido como palabra cuando su tamaño sea superior a 70 puntos.

Tras la aplicación de todo lo anterior y la extracción del camino correcto, nos aparece un resultado, el cual viene representado en la Figura 20:

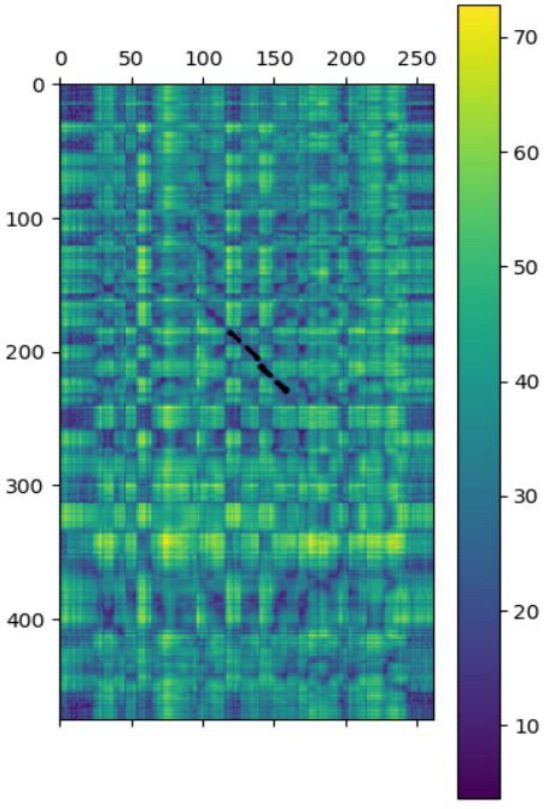

Figura 20. Matriz de distancias ejemplo (Resultado)

Las frases utilizadas en este caso de ejemplo son la siguientes:

1. Por favor puede ponerme con la **extensión** cero dos nueve uno.
2. Me pasaría con la **extensión** de Montoliu.

Como se puede comprobar, ha detectado en el centro de la matriz una secuencia de vectores que cumplen con el patrón y los cuales forman una palabra.

Tras este ejemplo se van a proceder a realizar distintas pruebas con nuevas frases para verificar como de correcto es el desarrollo.

4.5 Experimentación y resultados

De los doscientos casos de pruebas se van a utilizar los casos que tengan resultados más característicos y bien diferenciados para que se pueda ver diferentes casuísticas y comprobar que tal funciona en todos los ambientes posibles.

El ejemplo anterior se puede considerar el primero de los casos. En él se ha obtenido un único resultado correcto el cual coincide con la palabra que se debería haber detectado. Aquí no se ha detectado ruido ni se ha detectado ningún falso positivo. Pero esto no siempre es así.

4.5.1 Caso 1

El primer caso que va a ser analizado es a priori igual de sencillo que el anterior, se compararán dos frases muy similares entre sí y que además la palabra que se repite se encuentra aproximadamente en la misma posición del audio, tal y como sucedía en el caso anterior. Dichas frases son:

- Me gustaría **hablar** con Pablo Aibar por favor.
- Quiero **hablar** con la jefa de Juan Monfort.

Como podemos observar, la palabra que queremos que sea detectada es hablar y se encuentra a mitad de nuestros audios aproximadamente en la misma posición.

Paso 1.
Se realiza el cálculo de la distancia entre vectores y se obtiene su matriz de distancias (Figura 21):

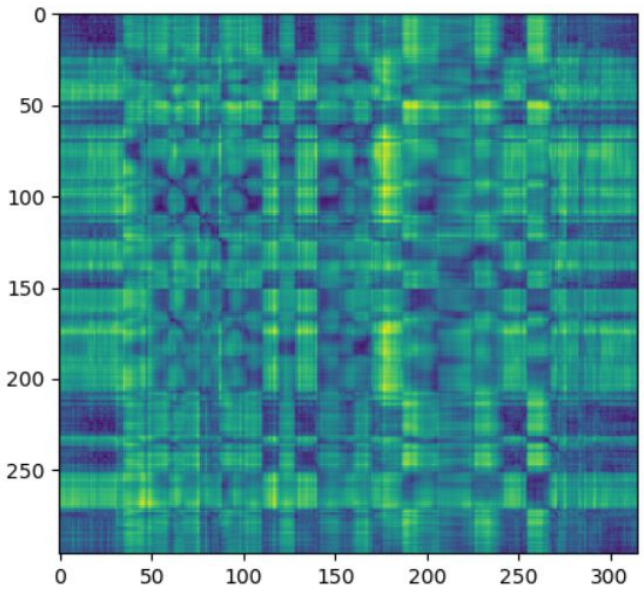

Figura 21. Matriz de distancias caso 1 (Representación)

Paso 2

Se aplica *thresholding* con un umbral de T1 = 15 / T2 = 25. (Figura 22)

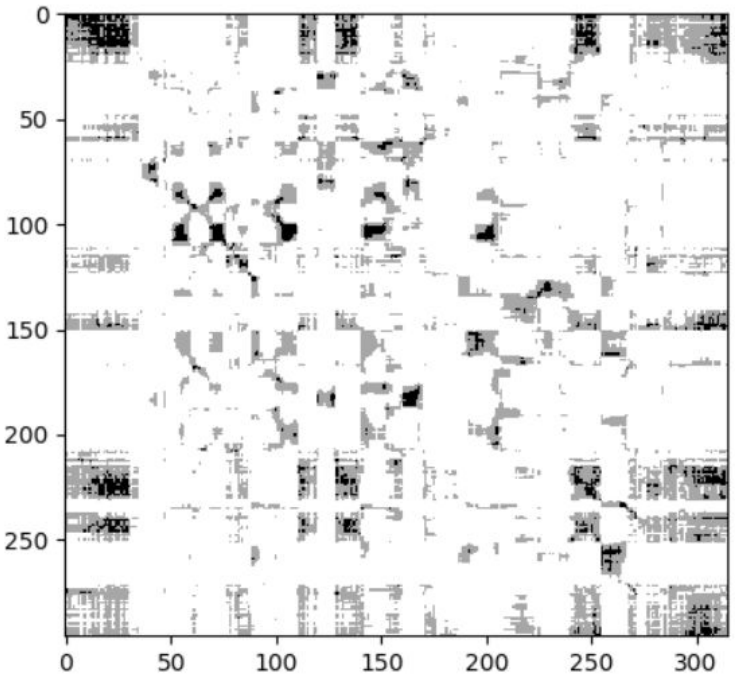

Figura 22. Matriz de distancias caso 1 (Thresholding)

Paso 3

Por último se hace el recorrido *DTW* para ver cuántos caminos, vectores de tamaño superior a 70 elementos, son encontrados. (Figura 23)

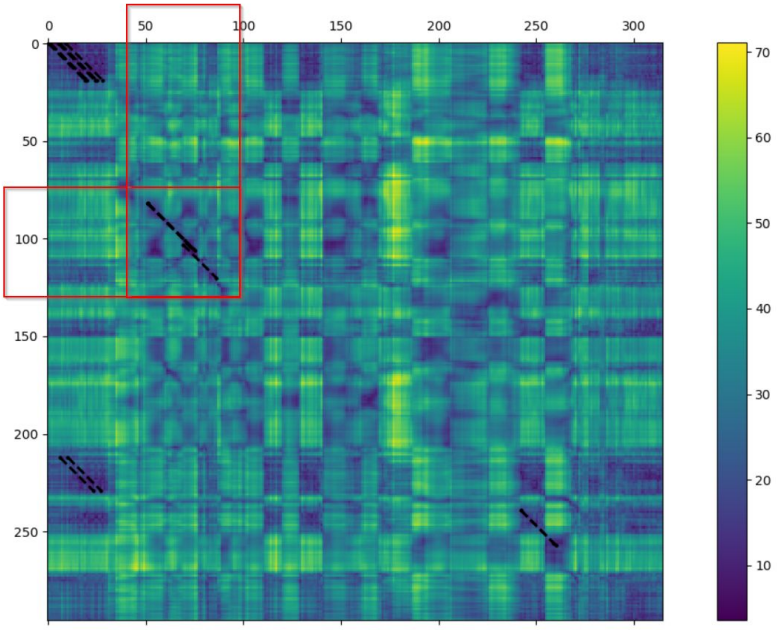

Figura 23. Matriz de distancias caso 1 (Resultado)

Como se puede observar en la imagen superior, hemos obtenido resultados positivos en cuatro sectores de nuestra matriz, pero el positivo más largo corresponde con la palabra “hablar” de cada uno de los audios dada la situación en que se encuentra. El resto son falsos positivos probablemente causados por el ruido al inicio y final de los audios, y por último un positivo en la parte inferior derecha. Dicho positivo puede ser causado por la similitud de fonemas entre “favor” y “Monfort”.

4.5.2 Caso 2

En este segundo caso la particularidad presentada se produce al tener dos segmentos o palabras que nuestro algoritmo tendrá que detectar, no solo uno.

- **Páseme con el** despacho de ana vilá **por favor**.
- **Páseme con el** número cuatro siete cero cinco **por favor**.

Son segmentos más largos y claramente diferenciados.

Paso 1
Distancia entre vectores, matriz de distancias principal(Figura 24):

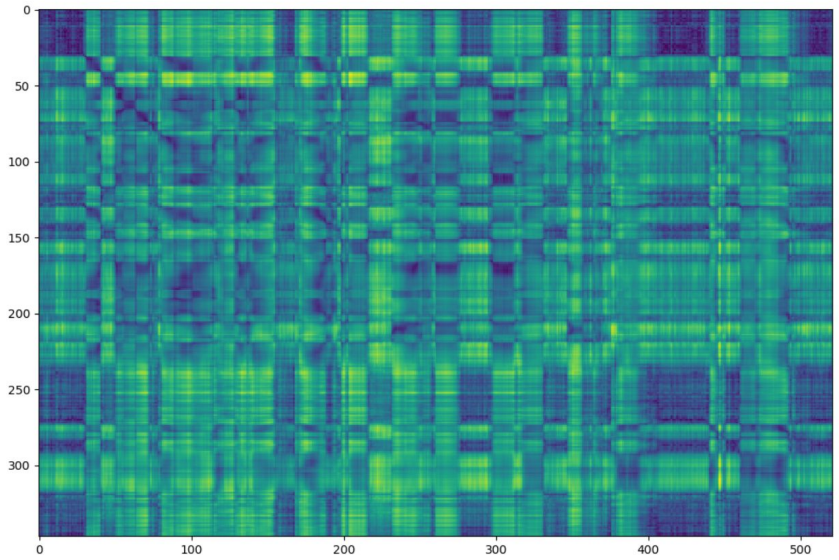

Figura 24. Matriz de distancias caso 2 (Representación)

Paso 2

Aplicamos *thresholding* con: T1 = 15 / T2 = 25 (Figura 25)

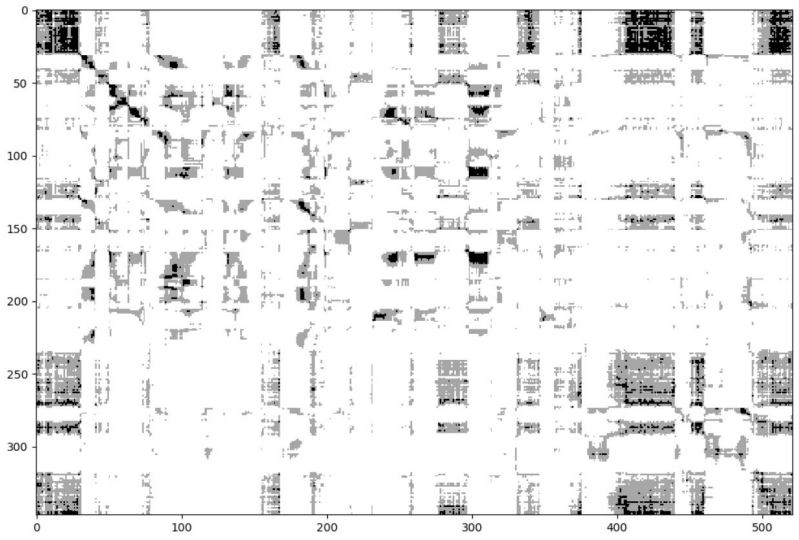

Figura 25. Matriz de distancias caso 2 (Thresholding)

Paso 3

DTW recogiendo vectores de tamaño superior a 80 elementos. (Figura 26)

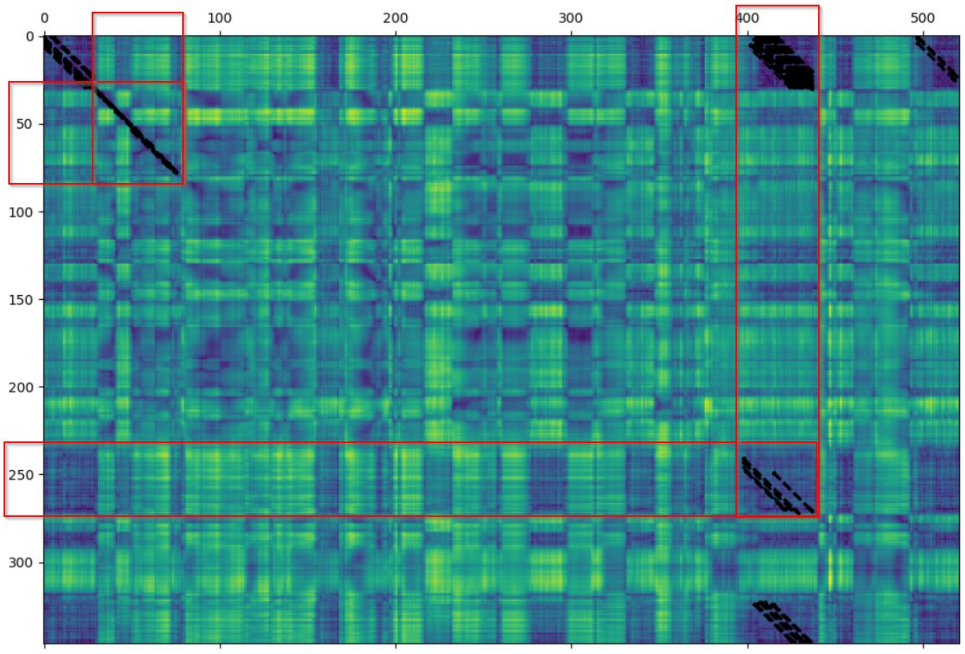

Figura 26. Matriz de distancias caso 2 (Resultados)

En este caso también se han encontrando ambos resultados. Un segmento muy largo al principio el cual hace referencia a “Páseme con el..” y llegando al final de ambos audios es encontrado el “por favor”. Esta solución si presenta un problema con el ruido en nuestras muestras. Se puede observar que siempre al inicio o final, es decir en los extremos de la matriz, encontramos una gran cantidad de caminos correctos los cuales se tendría que revisar y añadir como excepciones al algoritmo.

4.5.3 Caso 3

Tras ver los dos casos anteriores se puede apreciar que el algoritmo siempre detecta las soluciones correctas además de algunos falsos positivos, pero los ejemplos eran relativamente sencillos. En este tercer caso vamos a tratar de verificar que verdaderamente funciona complicando el proceso de búsqueda. Las frases a analizar son:

- **Por favor** puede ponerme **con la extensión** cero dos nueve uno.
- Me pasa **con la extensión** de sanz **por favor**.

En este caso particular se tienen dos segmentos, uno más largo y otro más corto, donde además las palabras a detectar están intercambiadas, es decir en el resultado se deberá apreciar cómo “por favor” aparece al principio del segmento largo pero al final del corto, y “con la extensión” aproximadamente por la mitad de ambos.

Paso 1
Distancia entre vectores, matriz de distancias principal (Figura 27) :

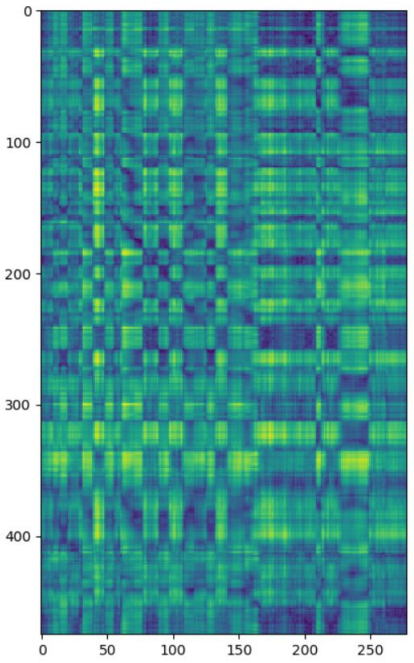

Figura 27. Matriz de distancias caso 3 (Representación)

Paso 2

Aplicamos thresholding con: T1 = 15 / T2 = 25 (Figura 28)

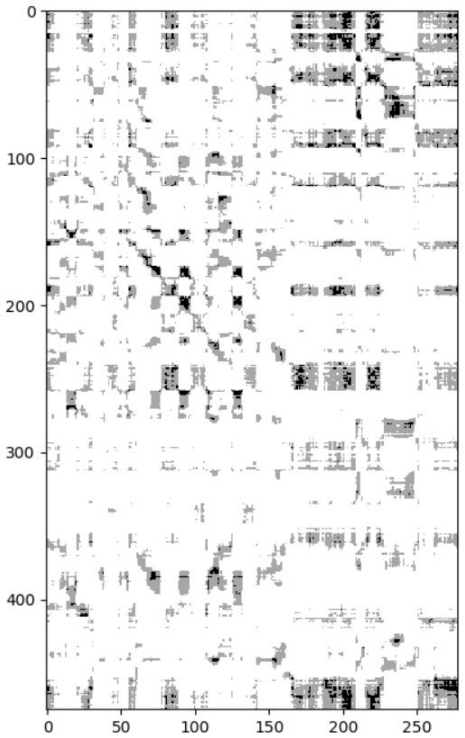

Figura 28. Matriz de distancias caso 3 (Thresholding)

Paso 3
DTW recogiendo vectores de tamaño superior a 80 elementos. (Figura 29)

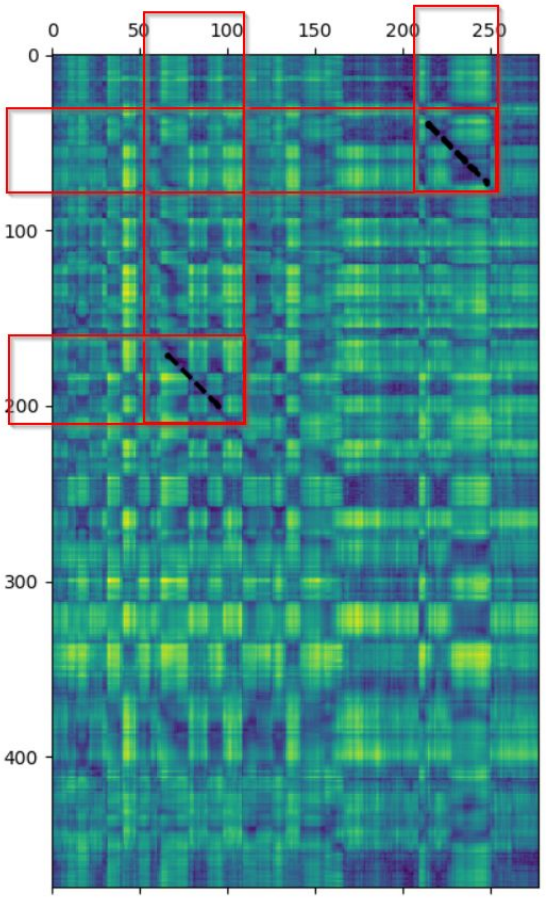

Figura 29. Matriz de distancias caso 3 (Resultado)

En este caso los resultados han sido muy precisos, solamente tenemos dos positivos.

El primero de los positivos se encuentra, tal y como se mencionó anteriormente, al final del audio corto pero al principio del audio largo (Arriba a la derecha) y hace referencia a “Por favor”. El segundo de los positivos se encuentra al principio/mitad de ambos audios y hace referencia a “con la extensión”. No se ha encontrado ningún falso positivo.

4.6 Conclusiones de las pruebas

Estos tres casos más el ejemplo con el que se ha explicado el proceso corresponden a los 4 casos principales que se han presentado durante la experimentación. Como se puede comprobar el algoritmo es bastante preciso ya que quitando el ruido del inicio y final de cada audio se puede verificar que en todos los casos ha detectado las palabras iguales tanto estando a inicios mitad o final de cada audio.

Capítulo 5
Conclusiones

Para finalizar el presente trabajo, vamos a comprobar hasta qué punto hemos alcanzado los objetivos que nos habían sido marcados anteriormente y se va a detallar cuáles podrían ser las mejoras, y trabajos futuros.

5.1 Conclusiones

El objetivo principal de este proyecto era el desarrollo de un sistema de procesamiento de lenguaje natural el cual permitiese la extracción de la información de un discurso hablado. Dicho discurso habrá sido previamente procesado pero no se dispondrá de ningún recurso propio del idioma a tratar, simulando así el proceso que un niño pueda realizar al dar los primeros pasos para aprender un idioma.

Con el fin de desarrollar el objetivo presentado y presentar una solución que se adapte a ello, han sido definidos dos problemas en dos escenarios diferentes. El primero en el capítulo tres, era el desarrollo de un sistema el cual a partir de la palabra contenida en un audio, fuera capaz de descubrir dicha palabra en un discurso largo.

Para ello se ha presentado el problema así como el material a utilizar, dos discursos largos y un conjunto de cien palabras diferentes. Para su resolución se ha decidido utilizar un híbrido entre los algoritmos de Park y Glass y Müller, donde la palabra completa era comparada con el discurso principal, creando una submatriz en cada iteración, recorriendo así punto a punto el audio largo.

Finalmente el resultado de este primer desarrollo era el punto inicial de cada una de las repeticiones de la palabra en el discurso. Dichos resultados mostraban un correcto funcionamiento del algoritmo a falta de una evaluación más exhaustiva, pero el proceso tenía un problema de eficiencia al tener un gran coste computacional.

En la segunda parte del proyecto se ha cambiado de escenario y ha sido presentado un proceso más complejo, donde como material se disponían de audios cortos de aproximadamente 10 segundos, con el objetivo de realizar la comparación de esos audios entre sí y extraer las palabras iguales que contuviesen.

Para su resolución, se ha hecho una adaptación del algoritmo *DTW* presentado por Muscariello, adicionando dos pasos previos al proceso principal del algoritmo. Durante el primero de los pasos se calculaba la distancia entre vectores de ambos audios, creando dicha matriz antes de empezar el desarrollo y usando como referencia cada punto de forma independiente sin hacer que guarden relación entre sí como haría de forma dinámica el *DTW*. En el segundo paso es añadido un proceso de *thresholding* definiendo dos umbrales, el primero más preciso para extraer los posibles puntos de inicio del algoritmo y el segundo un poco más laxo que el primero con el objetivo de acotar el espacio de movimiento de nuestro DTW y reducir el tiempo de ejecución.

Los resultados de esta segunda parte fueron bastante correctos. El análisis manual de los resultados confirmó que las palabras eran encontradas dentro de los audios en las posiciones esperadas. Adicionalmente el tiempo fué reducido de forma drástica, siendo aceptable dada la dificultad de la tarea.

5.2 Relación del trabajo desarrollado con los estudios cursados

Este trabajo está relacionado con diversas asignaturas, tanto genéricas de la carrera como de la rama de computación. Están presentes las asignaturas de algorítmica y estructura de datos y algoritmos durante el desarrollo de los distintos programas, pero como base del proyecto completo encontramos las asignaturas de aprendizaje automático y percepción, ya que son las que tocan el tema tratado durante el proyecto de forma más directa.

5.3 Trabajos futuros

A partir de las conclusiones extraídas en el apartado anterior, podemos comentar ciertos aspectos del desarrollo que podrían ser mejorados u otras aproximaciones que se podrían explotar:

- Evaluación de nuestros algoritmos. En ambos casos los resultados son analizados manualmente. Cuando se disponga de un corpus transcrito y etiquetado con marcas de tiempo, adem´s de un software de evaluación, podrá realizarse una evaluación objetiva que permita comparar los resultados con otras aproximaciones.
- Un segundo punto que se puede explorar en un futuro proyecto es aplicar y la representación basada en thresholding de la distancia entre frames usada en *Zero Resources Spoken Term Discovery* al problema de *Query by Example*.
- Por último, el principal trabajo que se podría continuar es seguir depurando nuestro desarrollo de DTW del segundo problema. Tratar de eliminar el ruido de inicio y final y realizar el testeo con audios más largos, más complejos y en otros idiomas.

Capítulo 6
Bibliografía

(Park y Glass, 2005) Park, Alex y James R Glass: *Towards unsupervised pattern discovery in speech*. En *Automatic Speech Recognition and Understanding*. Páginas 53–58, 2005.

(Priyanka, 2016) Priyanka A. Abhang, Bharti W. Gawali and Suresh C. Mehrotra: *Technological Basics of EEG Recording and Operation of Apparatus.* Páginas 10-25. 2016.

(Jansen et ál, 2011) Jansen, Aren y Benjamin Van Durme: Efficient Spoken Term Discovery Using Randomized Algorithms. En Automatic Speech Recognition and Understanding(ASRU), 2011 IEEE Workshopon, páginas 401–406. 2011.

(Gregory Hickok, 2016) Gregory Hickok, David Poeppel: *Neurobiology of Language*. Páginas 299-310. 2016.

(Muscariello et ál, 2012) Muscariello, Armando, Guillaume Gravier y Frédéric Bimbot: *Unsupervised Motif Acquisition in Speech via Seeded Discovery and Template Matching Combination.* Páginas 2030–2044, 2012.

(Muscariello et ál, 2009) Muscariello, Armando, Guillaume Graviery y Frédéric Bimbot: *Audio keyword extraction by unsupervised word discovery.* En INTERSPEECH 2009*: 10th Annual Conference of the International Speech Communication Association,* 2009.

(Muscariello et ál, 2011), Armando, Guillaume Gravier y Frédéric Bimbot: *Zero resource audio-only spoken term detection based on a combination of template matching techniques.* En *INTERSPEECH 2011: 12th Annual Conference of the International Speech Communication Association,* 2011.

(Rufiner et ál, 2004) Rufiner, Hugo L.; Milone, Diego H. Sistema de reconocimiento automático del habla. Ciencia, Docencia y Tecnología, vol. XV, núm. 28, mayo, 2004, pp. 151-177. Universidad Nacional de Entre Ríos . Argentina.

(Müller, 2007) Springer Berlin Heidelberg, Berlin, Heidelberg: *Information Retrieval for Music and Motion*, capítulo *Dynamic Time Warping*. Páginas 69–84, 2007.

(Jadwiga Rogowska, 2009)Jadwiga Rogowska: *Handbook of Medical Image Processing and Analysis (Second Edition)*. Página 73-90. 2009.

(Davies, 2012) E.R. Davies: *Computer and Machine Vision (Fourth Edition)*. Páginas 82-110 2012.

(Jadwiga, 2006) Rogowska Jadwiga PhD: *Optical Coherence Tomography, Principles and Applications*. Páginas 305-329, 329a. 2006.

Capítulo 7
Anexos

Publicaciones relacionadas

A continuación se enumeran las diferentes publicaciones relacionadas con el trabajo presentado:

- Sergio Laguna, Emilio Sanchis, Lluís-F, Hurtado and Fernando Garcia. ELiRF Query-by-Example STF systems for the ALbayzin 2016. Search on Speech Evaluation.
- F. García-Granada, E.Sanchís, M.J. Castro-Bleda, J.A.González, L.F.Hurtado. ZEROSPEECH2017 ELIFR-UPV SYSTEM.