Universidad Autónoma de Madrid

Escuela Politécnica Superior

Trabajo Fin de Grado

Detección de palabras clave en voz mediante ejemplos empleando redes neuronales profundas

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación (EUR-ACE®)

Autor: Quintela Gironás, Juan Carlos

Tutor: Torre Toledano, Doroteo

Departamento de Tecnología Electrónica y de las Comunicaciones

FECHA: Junio 2019

Detección de palabras clave en voz mediante ejemplos empleando redes neuronales profundas

Autor: Quintela Gironás. Juan Carlos

Tutor: Torre Toledano, Doroteo

Grupo Audias -Audio, Data Intelligence and Speech

Dpto. Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Junio 2019

Resumen

Actualmente nos vemos inmersos en un mundo donde los datos multimedia son cada vez más cuantiosos y frecuentes. Con el objetivo de extraer información y detectar palabras clave en ficheros de audio presentes en medios de comunicación e Internet, entre otras aplicaciones como la interacción con sistemas sin teclado o búsquedas para personas ciegas, surgen los sistemas QbE-STD (Query-by-Example Spoken Term Detecion).

Estos sistemas tienen como objetivo, buscar un ejemplo de un objeto o parte de él en otro objeto, que aplicado a nuestro trabajo consiste, en el reconocimiento de palabras o secuencias de ellas en archivos de audio.

En este Trabajo Fin de Grado se ha tomado como punto de partida el Trabajo Fin de Máster con el título de "Implementación y evaluación de un sistema QbE-STD (Query-by-Example Spoken Term Detection)" de María Cabello Aguilar con el fin de desarrollar un nuevo módulo, donde emplearemos redes neuronales profundas, que servirá para mejorar los resultados obtenidos en las últimas evaluaciones Albayzin 2016 y 2018 Search on Speech.

Al igual que anteriormente nuestro sistema deberá realizar la correcta detección independiente del idioma de la entrada o query, basada en términos hablados. Llegando incluso a ser posible que un usuario realice una búsqueda en un repositorio de audio emitiendo con su voz el término a buscar.

La técnica empleada para representar estos términos hablados ha sido la de posteriorgramas fonéticos. Estos posteriorgramas se han obtenido haciendo uso de los decodificadores fonéticos desarrollados por la Universidad de Tecnología de Brno (BUT), empleandose también el kit de herramientas de modelos ocultos de Markov oculto (HTK) para la correcta utilización de estos posteriorgramas.

Para realizar la detección de los terminos hablados en los correspondientes repositorios de audio se ha empleado las ya mencionadas redes neuronales profundas. Previo a esto se realizó un exahustivo trabajo de tratamiento de la base de datos con el fin de poder adaptar el material disponible a este nuevo módulo. De esta manera conseguimos desarrollar un sistema que puede servir como punto de partida para futuras vías de trabajo del grupo AUDIAS. [1]

Para el desarrollo de la solución y la realización de las pruebas se han utilizado los audios pertenecientes a las evaluaciones Albayzin 2016 y 2018 Search on Speech, mencionado anteriormente. Con la intención de obtener resultados que se puedan contrastar con otros sistemas publicados similares pudiendo llegar a ser competitivos y semejantes a los de otras implementaciones parecidas.

Palabras clave

Datos multimedia, extracción de información, detección de palabras clave, sistemas Query-by-Example Spoken Term Detecion (QbE-STD), reconocimiento de palabras, redes neuronales profundas, posteriorgramas, tratamiento de base de datos.

[1] http://audias.ii.uam.es/

Abstract

Nowadays we are currently immersed in a world where multimedia data are increasingly numerous and frequent. With the aim of extracting information and detecting key words in audio files present in the media and the Internet, among other applications such as interaction with keyboardless systems or searches for blind people, QbE-STD (Query-by-Example Spoken Term Detecion) systems emerge.

These systems aim to find an example of an object or part of it in another object, which applied to our work consists in the recognition of words or sequences of them in audio files.

In this Final Degree Project, the Master's Thesis has been taken as a starting point with the title "Implementation and evaluation of a QbE-STD system (Query-by-Example Spoken Term Detection)" by María Cabello Aguilar in order to develop a new module, where we will use deep neural networks to improve the results obtained in the latest evaluations Albayzin 2016 and 2018 Search on Speech.

As before, our system must perform the correct detection regardless of the language of the input or query, based on spoken terms. It is even possible for a user to perform a search in an audio repository by saying the search term with his voice.

The technique used to represent these spoken terms has been phonetic posteriorgrams. These posteriorgrams have been obtained using the improved phonetic decoders of the University of Technology of Brno (BUT), using also the tool kit of the hidden Markov model (HTK) for the correct use of these posteriorgramasgrams.

To perform the detection of spoken terms in the corresponding audio repositories, have been used the aforementioned deep neural networks. Prior to this, an exhaustive database treatment work was carried out in order to adapt the available material to this new module. In this way we are able to develop a system that can serve as a starting point for future work paths of the AUDIAS[2] group.

For the development of the solution and the realization of the tests, the audios belonging to the evaluations Albayzin 2016 and 2018 Search on Speech, mentioned above, have been used. With the intention of obtaining results that can be compared with other similar published systems, being able to become competitive and similar to those of other similar implementations.

Keywords

Multimedia data, information extraction, keywords detection, Query-by-Example Spoken Term Detecion (QbE-STD) systems, word recognition, deep neural networks, posteriorgrams, database treatment.

[2] http://audias.ii.uam.es/

Agradecimientos

A mis padres y a mis hermanas, por estar apoyándome constantemente en esta etapa tan intensa y exigente. Sin vosotros no sería quién soy ni estaría donde estoy.

A mi grupo de amigos más cercano que me ayudaron a centrarme y a dar todo de mí en los momentos más complicados.

A mi tutor Doroteo, que fue de gran ayuda y apoyo en el desarrollo de este trabajo.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción.

1.1 Motivación.

Como es sabido, nos encontramos en una época en la que la cantidad de datos que se manejan son realmente grandes, y crecen de una manera exponencial. Esto afecta por supuesto también a los contenidos multimedia, haciendo necesario una manera de poder trabajar con estos datos y acceder a partes de ellos de una forma rápida, eficiente y fiable.

Esta necesidad mencionada desencadena de forma más concreta en el desarrollo de un sistema con el que poder tratar la información de audio de manera más específica, teniendo en el punto de mira el reconocimiento de habla y el de palabras clave, siendo este segundo donde podemos encontrar distintas técnicas como el QbE (Query by example) que es sencillamente la búsqueda de información en grandes repositorios empleando ejemplos como el fragmento a buscar y STD (spoken term detection) que permite que estos ejemplos sean términos hablados producidos por personas.

Con estas dos técnicas mencionadas anteriormente surge la fusión de ambas en los sistemas conocidos como QbE-STD, sistema que se desarrolló en gran medida en el Trabajo Fin de Máster del que este TFG es continuación.

Del mismo modo que hemos presentado estas técnicas se hace necesario hacer lo mismo con el método utilizado a lo largo de este proyecto con el fin optimizar la manera en la que se encontraban las coincidencias entre queries y repositorios, este método se basa en las consolidadas redes neuronales profundas, empleadas en el aprendizaje automático.

Gracias a las distintas evaluaciones existentes de sistemas de QbE-STD hemos podido emplear bases de datos realmente competentes y con las que poder saber la corrección y nivel de nuestro sistema. En concreto hemos empleado la evaluación Albazyn Search on Speech, en la que la universidad participa de forma activa.

Con todo esto, el objetivo principal de este Trabajo de Fin de Grado es llegar a implementar un sistema QbE-STD empleando las redes neuronales profundas mencionadas con anterioridad. Pudiendo mejorar todo lo posible los resultados obtenidos en la pasada evaluación y de esta manera poder sentar las bases para trabajos futuros.

Para poder alcanzar el objetivo marcado se ha realizado un trabajo previo de tratamiento de datos para poder acomodarlos a las necesidades de nuestro sistema. También se han empleado técnicas aprendidas a lo largo del último curso en la asignatura de Tratamiento de Señales de Voz y Audio.

1.2 Objetivos.

Con la finalidad de alcanzar el objetivo final de este Trabajo de Fin de Grado que es el desarrollo de un sistema capaz de detectar palabras clave de forma rápida y precisa, previamente establecimos distintos objetivos intermedios según se realizaba el desarrollo del sistema. Estos objetivos parciales han sido:

- Familiarización con los datos que se emplearán en la práctica con el fin de extraer parámetros necesarios para la implementación del sistema.

- Tratamiento de los datos para adaptarlos a las necesidades de nuestro nuevo sistema y poder extraer características de ellos, útiles para facilitar la detección de los términos.

- Desarrollo de la red neuronal que se empleará en el proceso de aprendizaje automático y que más adelante será la encargada de proporcionarnos las coincidencias.

- Llevar a cabo la evaluación propuesta por Albazyn Search on Speech para conocer de forma objetiva los logros alcanzados.

- Tratar de añadir nuevas aportaciones personales siempre que sea posible y así conseguir diferenciarnos de sistemas similares.

A continuación, se puede observar un diagrama donde se ve la duración en días que implicó cada una de las partes en las que se dividió el proyecto para alcanzar cada uno de los objetivos parciales para desembocar en el resultado final.

Actividad	1-30 nov	1-31 dic	1-31 en	1-28 feb	1-31 mar	1-30 abr	1-31 may	1-20 jun
Documentación y análisis de BBDD								
Tratamiento y adaptación de datos								
Desarrollo del sistema								
Experimentos y pruebas								
Memoria								

Figura 1 – Cronograma de las tareas realizadas en el TFG

1.3 Organización de la memoria.

La presente memoria de este Trabajo Fin de Grado tendrá la siguiente organización y estructura:

- **Capítulo 1: Introducción.**

  En este apartado presentaremos de forma general en lo que consiste este Trabajo Fin de Grado. Definiremos las motivaciones que han llevado a su realización, así como los objetivos que tenemos con el mismo.

- **Capítulo 2: Estado del arte.**

  Aquí realizaremos una exhaustiva exposición de aquellos detalles y conceptos teóricos necesarios para la comprensión del sistema desarrollado. Es decir, detallaremos las partes que lo componen, así como los métodos y técnicas que se utilizan actualmente en ellos. Del mismo modo, se explicará de forma clara los términos específicos usados en todo el documento.

- **Capítulo 3: Diseño y desarrollo.**

  Este apartado servirá para detallar los diferentes elementos que conforman nuestro entorno experimental. Se realizará una descripción de los distintos aspectos relevantes en el desarrollo del sistema como son la base de datos empleada, así como las tecnologías utilizadas en la fase de reconocimiento, explicando las distintas fases que conforman nuestro sistema final y la evaluación que se llevará a cabo.

- **Capítulo 4: Pruebas y resultados.**

  En este capítulo se expondrán los distintos experimentos llevados a cabo, con el fin de llegar a los mejores resultados posibles. También se mostrarán las evaluaciones llevadas a cabo para determinar de forma objetiva los resultados del sistema desarrollado.

- **Capítulo 5: Conclusiones y trabajos futuros.**

  En este último apartado se realizará una valoración del trabajo realizado y obtenido, junto con unas conclusiones tratando de evaluar de forma objetiva el resultado final. Por último, se mostrarán posibles líneas de trabajo de cara al futuro relacionados con el sistema elaborado.

Capítulo 2

Estado del arte.

2.1 Introducción a los sistemas QbE-STD y las redes neuronales.

Las técnicas empleadas en la recuperación y búsqueda de información de audio son bastante diversas. Como ya hemos mencionado anteriormente nos centraremos en la detección de términos hablados (Spoken Term Detection, STD). Del mismo modo hay que destacar la utilización de la consulta por ejemplo (Query by Example, QbE). Y finalmente las redes neuronales que adquieren protagonismo siendo el método empleado para la final decisión de detección.

Comencemos hablando de **Query by Example (QbE)**. Este es un método realmente extendido en distintos ámbitos y desde luego el audio no se queda atrás. QbE puede entenderse como un lenguaje de consulta creado por Moshe Zloof de IBM en 1970 que permite al usuario hacer la búsqueda de archivos (texto, audio, imágenes, etc.) basándose en un ejemplo que puede ser parte del repositorio donde se realizará la búsqueda o una parte nueva introducida por el usuario que pretende encontrar por similitud [1]. A continuación, se puede ver un esquema típico de esta técnica.

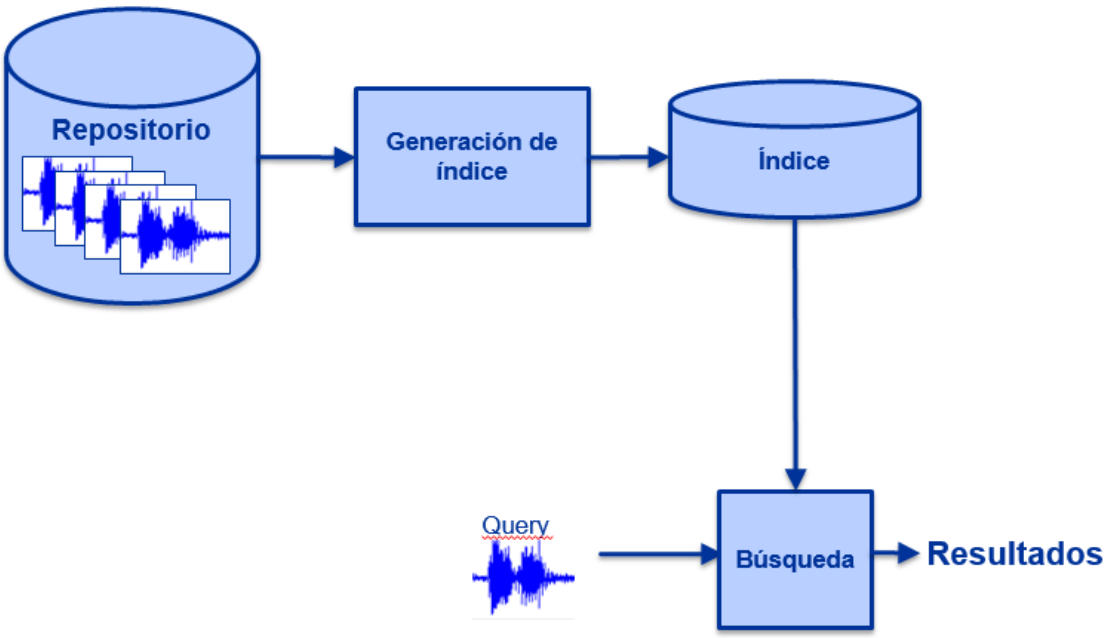

Figura 2 - Diagrama de bloques de un sistema QbE [2]

Spoken Term Detection (STD) añade a lo visto anteriormente que, además de tener como finalidad encontrar palabras o sucesión de palabras clave en un repositorio, la entrada al sistema es un término hablado ya sea en formato de audio o de texto, siendo este segundo bastante más común. Estos sistemas indican, aparte de las ocurrencias de cada una de las queries, el documento donde se encontró y el tiempo de comienzo y fin de esta ocurrencia admitiéndose un margen de error en esta estimación de tiempos, aproximadamente de 0.5 segundos [20]. A continuación, se muestra un esquema típico de un sistema STD.

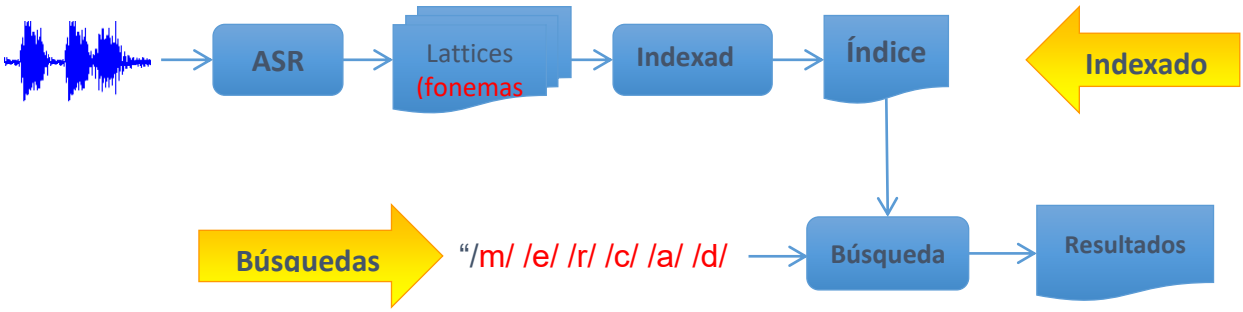

Figura 3 - Diagrama de bloques de un sistema STD [2]

Cabe destacar de estos sistemas su manera de evaluación siendo la más común el Actual Term Weighted Value (ATWV) que podríamos definir de la siguiente manera, para una query [20]:

$$ATWV = \frac{H}{N} - \beta \frac{FA}{T - N} \qquad \beta = 999.9$$

Donde:

H es el número de detecciones correctas

FA es el número de falsas alarmas

N es el número de veces reales que aparece la query

T es el tiempo total del audio donde hemos buscado (en segundos)

β es un factor de ponderación

El objetivo es maximizar este valor de ATWV, siendo 1.0 el valor más grande correspondiente a un sistema perfecto. Este método depende de la elección del umbral es por ello que se utiliza el MTWV (Maximum Term Weighted Value) que no depende de esta elección de umbral [20].

Redes neuronales profundas este será el bloque de decisión, el cual nos dirá a su salida si la muestra analizada corresponde con una ocurrencia positiva o negativa. Del mismo modo determinará donde se encuentra esta ocurrencia en caso de ser positiva, datos necesarios para realizar la evaluación.

2.2 Descripción de un sistema QbE-STD con redes neuronales profundas.

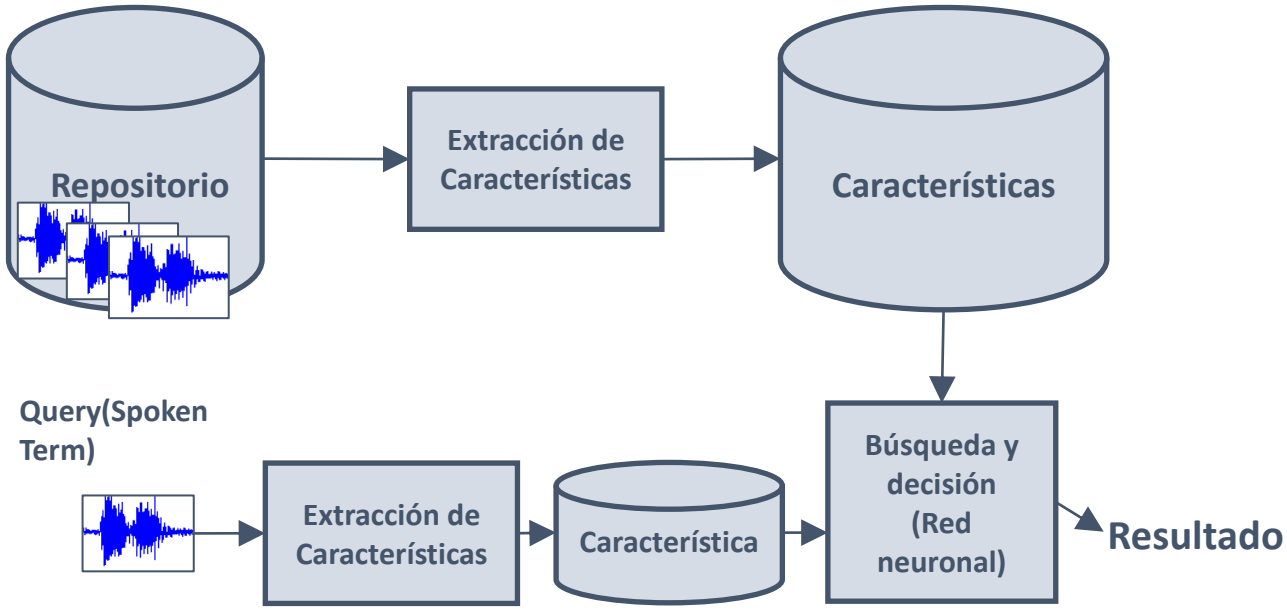

Figura 4 - Diagrama de bloques de un sistema QbE-STD fusioado con Red neuronal como sistema de decisión [2]

En la figura anterior puede observarse un esquema básico de un sistema QbE-STD fusionado con una red neuronal profunda de decisión. Las partes principales que tenemos en este sistema son las siguientes:

- **Repositorio.**

  Es el archivo o conjunto de archivos donde se quiere detectar la ocurrencia dada por la query.

- **Query (Spoken Term).**

  Es la consulta en forma de término hablado que queremos encontrar en nuestra base de datos determinada por el repositorio.

- **Extracción de características.**

  Como se puede apreciar esta etapa es común en ambas ramas ya que juega un papel crucial a la hora de realizar la detección más adelante. La finalidad de esta fase no es más que obtener una serie de características que puedan ser relevantes y determinantes en procesos contiguos. Esta etapa sufrirá cambios en función de si la query se encuentra en formato de audio a la entrada o en formato de texto. El primer caso es el empleado en este trabajo y cabe destacar que en este apartado se determina de manera probabilística la existencia de un fonema u otro

siendo estas las características que se emplearán en el reconocimiento. Esto se explicará de manera más detallada en la sección 2.2.1.4.

- **Características.**

  Esta fase trata de realizar una organización de los datos con la finalidad de poder realizar búsquedas más eficientes.

- **Búsqueda y decisión (red neuronal profunda).**

  Esta etapa presenta la estructura de una red neuronal que será la encargada de realizar la búsqueda y de decidir si se encuentra una ocurrencia positiva o negativa. Esta red deberá aprender a partir de una serie de datos de entrenamiento a realizar este proceso de forma automática.

- **Resultados.**

  Como salida de este sistema obtendremos lo que denominaremos hipótesis de ocurrencias. Aquí se recogerá la parte del repositorio donde se prevé que se encuentra la query.

Con esto podemos centrarnos de forma más detallada en, posiblemente, el bloque más importante de la cadena anteriormente vista. Nos referimos a la **Extracción de características.**

2.2.1 Bloque de Extracción de características.

Hay diversas técnicas que nos permiten realizar una extracción de características de audio con el fin de poder emplearlas para el reconocimiento de habla, palabras clave, etc. Entre ellas destacaremos los coeficientes PLP (Predicción Lineal Perceptual), LPC (Coeficientes de Predicción Lineal), MFCC (Mel-Frecuency Cepstral Coefficients) y posteriorgramas fonéticos (PPGs), aunque se pueden emplear muchos otros más.

2.2.1.1 Coeficientes PLP.

Comencemos explicando el concepto de PLP (Predicción Lineal Perceptual). Esta es una técnica que surge de la combinación de la LP (Predicción Lineal) y la DFT (Transformada Discreta de Fourier). PLP modela el discurso humano basado en el concepto de psicofísica de la audición, trata de descartar la información menos importante con el fin de mejorar el reconocimiento. Su forma de cálculo es igual a la de los LPC excepto en la transformación de sus características espectrales [8]. Con esta técnica se realiza la aproximación de tres aspectos perceptivos de gran importancia como son las curvas de resolución de las bandas críticas, la curva de sensación sonora

(loudness) y la relación entre intensidad real y percibida quedando un esquema como el siguiente:

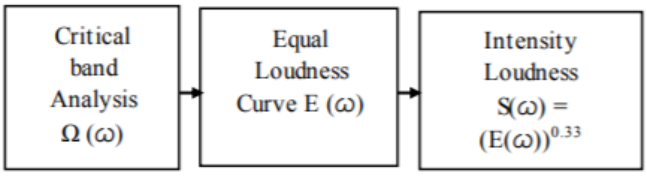

Figura 5 - Diagrama de bloques de procesamiento de diagramas PLP [8]

2.2.1.2 Coeficientes LPC.

Son una herramienta empleada principalmente en el procesamiento de la señal de audio y el procesamiento de la voz para representar la envolvente espectral de una señal digital de voz.

Este análisis LPC nos permite obtener una señal residual a partir de una trama de voz, la cual se corresponde con la excitación vocal (estructura fina espectral), es por ello que nos permite realizar una estimación del pitch y de la estructura armónica de alta precision [8].

2.2.1.3 Coeficientes MFCC.

Estos son los coeficientes derivados del análisis sobre la escala de Mel. Es el método dominante y más empleado en el reconocimiento de voz basandose en la escala perceptiva del oído humano. Esto se debe a la gran precision que presentan respecto a características temporales [8].

El procedimiento de cálculo de forma simplificada es el siguiente [21]:

1. Filtrado de pre-énfasis para las frecuencias más altas que se encuentran atenuadas y enventanado de la señal empleando la ventana de Hamming.
2. Transformada de Fourier emplenado el algoritmo FFT para reducir el coste operacional.
3. Aplicación de un banco de filtros Mel que emulan las bandas críticas.
4. Cálculo de la energía total logarítmica para cada banda.
5. Aplicación de la DCT/IFFT, con el fin de obtener la respuesta en el dominio cepstral, de la salida del bloque anterior.

2.2.1.4 Posteriorgramas fonéticos (PPGs).

Los **posteriorgramas fonéticos** son una matriz de clase frente a tiempo representando las probabilidades de cada clase fonética para cada instante de tiempo. Una clase

fonética puede referirse a un fonema, un fonema y su contexto fonético (fonemas anterior y posterior) [9].

Estos posteriorgramas se obtienen haciendo uso de reconocedores fonéticos. Gracias a estas matrices podemos obtener un tipo de características que recogen las probabilidades que se asignan para cada uno de los fonemas posibles. Como hemos mencionado puede referirse a un fonema, constar por ejemplo de 3 partes donde se recogen las probabilidades para cada uno de estos fonemas en cada uno de los instantes de inicio, mitad y fin del fonema.

A continuación, se puede observar un ejemplo de posteriorgrama donde el eje X representa el tiempo en segundos y el eje Y cada uno de los índices de los fonemas posibles:

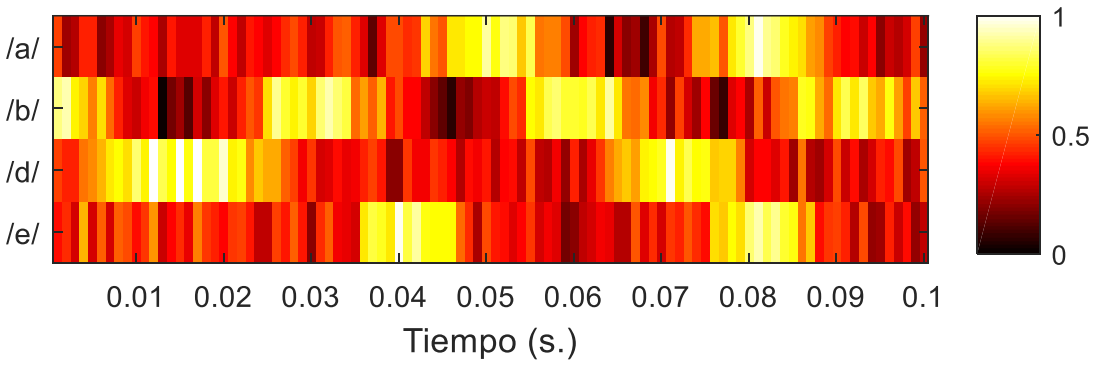

Figura 6 – Ejemplo de posteriorgrama fonético [2]

2.2.2 Métodos de reconocimiento.

El reconocimiento de audio se ha convertido en una de las principales preocupaciones de la tecnología. Ya que un correcto modelado del lenguaje puede tener multitud de aplicaciones como búsqueda mediante la voz, traducción directa, establecer órdenes a sistemas remotos, etc.

Esto ha llevado al desarrollo de diferentes técnicas con las que poder facilitar esta tarea, pudiendo evolucionar en ellas hasta lograr mejoras significativas de eficiencia computacional. En este apartado realizaremos un análisis concreto de los métodos más empleados actualmente como son DTW (Alineamiento Temporal Dinámico y sus modificaciones), HMMs (Modelos Ocultos de Markov) y las cada vez más empleadas redes neuronales.

2.2.2.1 DTW y sus variantes.

Es de interés analizarlo ya que el sistema del que partimos emplea este método. **DTW** o alineamiento temporal dinámico hace referencia a la técnica la cual surge debido a la problemática inherente a las diferentes formas de realizar una misma locución [26]. Esta técnica consiste en la realización de un alineamiento interno con el fin de poder realizar comparaciones coherentes entre patrones ya que es posible que las unidades fónicas a comparar, sin realizar este alineamiento, no se correspondan dando lugar a resultados sin sentido y erróneos.

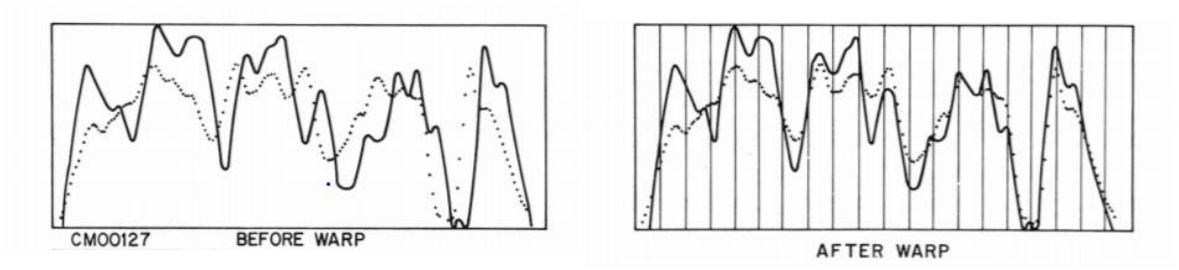

Figura 7 – Señal previa al alineamiento temporal (izquierda) y tras el alineamiento (derecha) [26]

A continuación, veremos de forma simplificada la manera de proceder para lograr este alineamiento temporal.

Nuestro objetivo es lograr la denominada función de alineamiento que es la que nos permitirá realizar el proceso. Si partimos de dos patrones o vectores de características $A=\{a_1, a_2, \dots, a_M\}$ y $B=\{b_1, b_2, \dots, b_M\}$ relacionados ambos por medio de C={ c(1), c(2), … , c(k)} donde cada *c(k)* es un puntero a los elementos a comparar, debemos establecer para cada *c(k)* una función de coste *d{c(k})* la cual puede ser por ejemplo la distancia euclídea $d(c(k))=(a_N - b_M)^2$. Llegando de esta manera a la función de alineamiento antes mencionada que será aquella que minimice la función de coste total $D(C) = \sum_{k=1}^{K} d\{c(k)\}$ *[26]*.

Cabe destacar que este alineamiento que realizamos no es lineal y que debe cumplir una serie de restricciones como son la de pendiente o la de ventana de ajuste limitando de esta manea la forma de la función de alineamiento.

El proceso de cálculo resumido anteriormente se resuelve haciendo uso de la programación dinámica, y podemos obtener como resultados las llamadas matrices de coste y de coste acumulado donde podemos ver el camino de alineamiento óptimo como apreciamos en la figura siguiente.

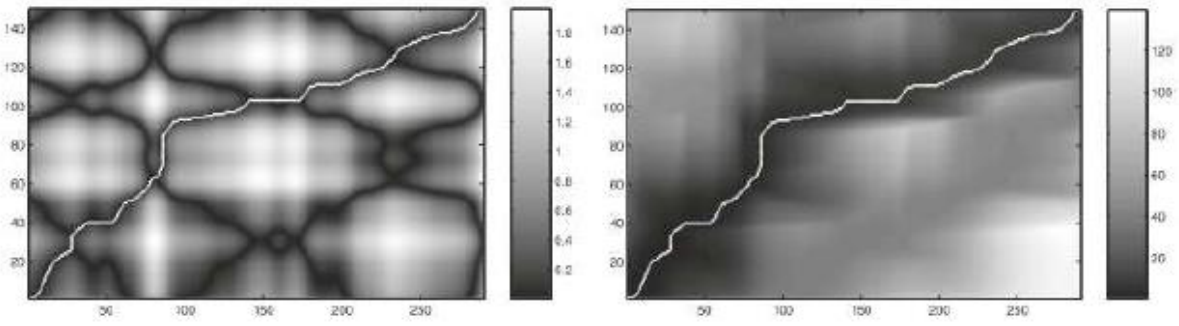

Figura 8 – Matriz de coste (izquierda), matriz de coste acumulado (derecha), señalando en ambas el camino óptimo [27]

Como variaciones a este algoritmo de DTW hay que destacar dos S-DTW (DTW de Subsecuencias) y Ms-DTW (DTW Multiescala) cuya función es similar a la vista en DTW, pero cambian las circunstancias de uso.

S-DTW

Esta técnica es una variación de la descrita anteriormente que se aplica en aquellos casos en los que los patrones a comparar presentan tamaños temporales muy distintos.

La principal modificación es que en este caso no podemos realizar un alineamiento de la duración total de ambos vectores de características ya que las comparaciones carecerían de sentido, es por ello que la forma de actuar en este caso es la de buscar una subsecuencia dentro de la secuencia más larga que se parezca lo más posible a la secuencia más corta. De esta manera podremos realizar un alineamiento interno para secuencia corta y subsecuencia pudiendo realizar comparaciones coherentes [4] [11].

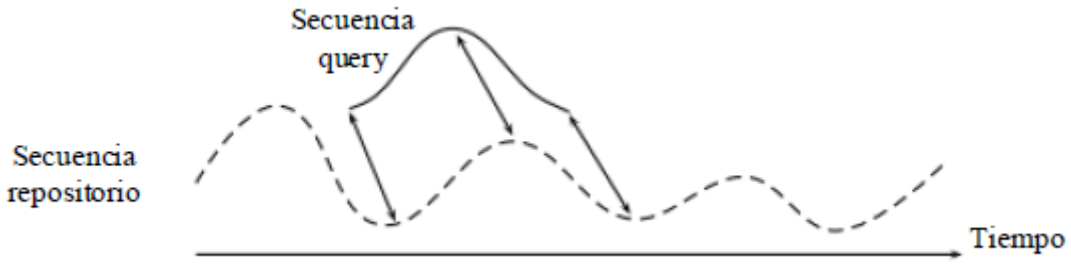

Figura 9 – Alineamiento entre query y subsecuencia del repositorio de búsqueda [27]

Ms-DTW

MsDTW tiene como objetivo reducir los requisitos computacionales de DTW calculando primero una alineación en un nivel de resolución de característica general [10].

La alineación aproximada en un nivel superior se proyecta en un nivel de resolución cada vez más fino. Su forma de actuar se asemeja a lo que vemos en la siguiente figura.

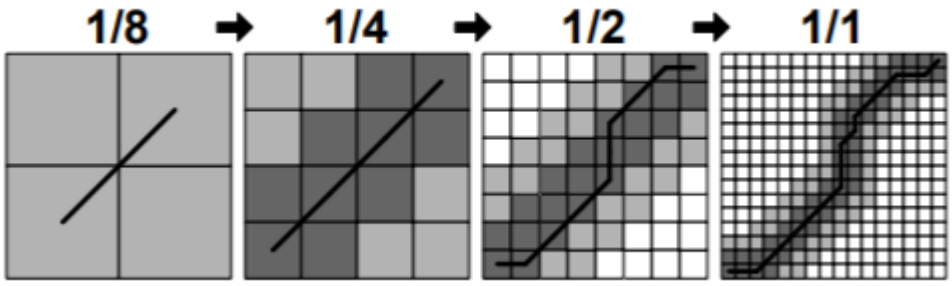

Figura 10 – Resoluciones empleadas en Ms-DTW [10]

2.2.2.2 HMMs (Modelos Ocultos de Markov).

Los Modelos Ocultos de Markov son la técnica más empleada en reconocimiento de voz. Estos modelan de forma estadística la acústica de la voz en lugar de forma determinista como lo hace DTW [12].

Surgen ya que en numerosas aplicaciones de reconocimiento del habla nos interesa poder predecir el contenido de la señal acústica. Por ello los reconocedores pretenden

determinar la secuencia de fonemas, definidos como estados, que formaron la palabra o la secuencia de la señal de voz.

Por tanto, podemos destacar que estos modelos están formados por un número de estados finito, y podemos pasar de uno a otro teniendo en cuenta las probabilidades de transición, las cuales se definen en una matriz con dicho nombre. Del mismo modo cuando nos encontramos en algún estado, estos presentan una probabilidad de observación y un número de símbolos observables, siendo sólo observable el resultado, quedando los estados "ocultos" [12].

Los Modelos de Bakis o de izquierda a derecha, se caracterizan porque una vez salimos de un estado no podemos volver a él, por lo que son bastante buenos a la hora de modelar señales que varían con el tiempo como es la voz, ya que las personas una vez terminamos de pronunciar un fonema no volvemos al mismo, sino que pasamos a otro, aunque si podemos pasar a otra realización del mismo fonema [12].

2.2.2.3 Redes neuronales.

Las redes neuronales artificiales han sido y son actualmente un área de gran importancia y estudio dentro de la Inteligencia Artificial. Estas tratan de imitar el comportamiento del cerebro humano con el fin de desarrollar sistemas capaces de resolver de forma automática problemas de gran dificultad, aplicando la experiencia y el aprendizaje.

Estas redes se estructuran inspirándose en la forma en que lo hace el cerebro, contando con distintos bloques llamados neuronas que se encuentran interconectadas creando estructuras similares a la siguiente:

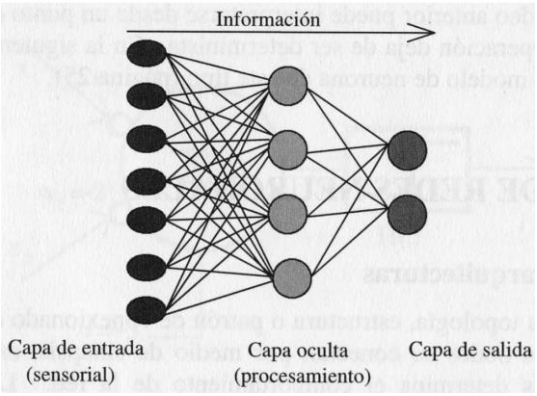

Figura 11 – Estructura típica de red neuronal [13]

Una vez definida de forma genérica lo que son las redes neuronales vamos a centrarnos en diferentes características que nos aportan.

2.2.2.3.1 Elementos Básicos.

A continuación, podemos ver un esquema de una RNA totalmente conectada:

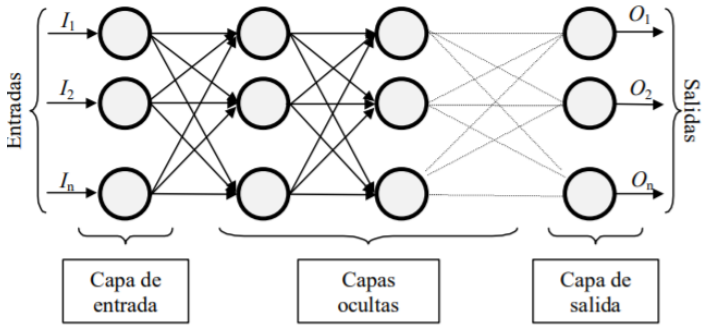

Figura 12 – Esquema red neuronal full-connected [13]

Como se puede apreciar podemos distinguir tres grandes capas. En primer lugar, la capa de entrada o también denominada capa sensorial es por donde ingresan los datos. Las capas ocultas son las que no presentan una conexión directa con su entorno y se encargan de realizar las operaciones o tareas oportunas de la red. La capa de salida que se encarga de proporcionar los resultados obtenidos por el trabajo realizado en las capas anteriores [13].

Todas estas capas comparten que están formados por el mismo bloque, neuronas. Cada una de estas neuronas corresponde a una unidad de procesamiento de información. Su funcionamiento es simple, estas reciben desde las neuronas de sus capas anteriores la información necesaria para posteriormente generar un resultado que se transmitirá a las neuronas siguientes con las que está conectada si cumple cierto criterio como puede ser un umbral [14].

A continuación, se puede ver la estructura básica de una neurona artificial. Esta opera con los valores recibidos teniendo en cuenta los pesos (W) de los enlaces correspondientes dando más importancia a los pesos mayores. Como hemos mencionado anteriormente si el resultado de la función F de la neurona es mayor que un valor umbral (U), la neurona se activa y emite una señal hacia las neuronas de la capa siguiente. En caso contrario la neurona permanece inactiva y no envía ninguna señal.

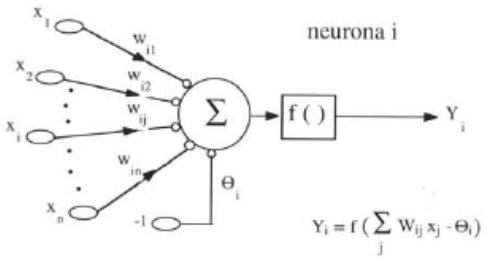

Figura 13 – Estructura típica de una neurona [14]

2.2.2.3.2 Funciones de entrada y activación.

Hemos definido en el apartado anterior la estructura básica de una neurona. Debemos además destacar las funciones que lleva a cabo.

Función de entrada

Esta función nos permite combinar las diferentes entradas de la neurona en una única entrada global teniendo en consideración los pesos de las conexiones por las que vino cada entrada independientemente [13].

Algunos ejemplos de funciones de entrada pueden ser el sumatorio de todas las entradas o el máximo de entre todas las entradas.

Función de activación

Esta trata de emular a la neurona biológica la cual puede estar activa o inactiva. Una de las principales diferencias es que la neurona artificial puede tener diferentes estados de activación (no solo dos) [13].

Con esta función lo que tratamos de determinar es cuánto de activa se encuentra la neurona. Para ello realiza una transformación de la entrada conjunta anteriormente obtenida en un valor de activación.

Podemos destacar principalmente 2 funciones de activación:

- *Función sigmoidea*. Es la más empleada históricamente y se denota por la siguiente ecuación:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  Aquellas neuronas que emplean esta función reciben el nombre de neuronas sigmoideas. Esta función reduce nuestra salida al rango de 0 a 1 [13].

- *Función ReLU (Rectified Linear Unit).* Esta se ha convertido en la más empleada actualmente debido principalmente en que la sigmoide dificulta el entrenamiento para redes de muchas capas [13]. La función ReLU se define de una manera mucho más sencilla:

$$R(z)=max(0,z)$$

  Como podemos apreciar las ReLUs actúan dejando pasar lo valores positivos sin cambiarlos, y poniendo a 0 los valores negativos. A continuación, se aprecia la forma de ambas funciones.

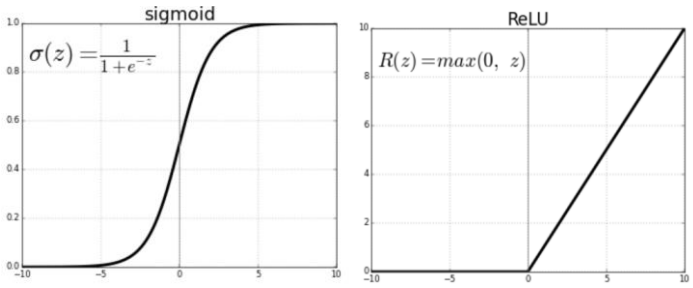

Figura 14 – Funciones de activación sigmoid y ReLU [13]

Hay muchas más funciones de activación, sin embargo, las más importantes y empleadas son las descritas. Podríamos destacar también la función lineal o la tangente hiperbólica que presentan las siguientes formas respectivamente:

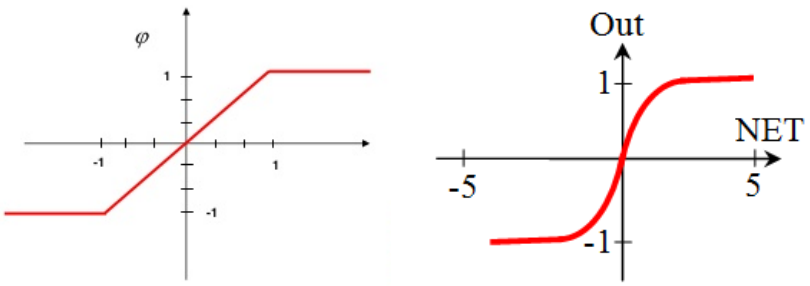

Figura 15 – Funciones de activación lineal(izquierda) y tangente hiperbólica(derecha) [31][32]

2.2.2.3.3 Tipos de aprendizajes.

Las redes neuronales, como hemos mencionado anteriormente, requieren de una etapa de entrenamiento donde poder adquirir el aprendizaje por medio de la experiencia que más tarde emplearán. Sin embargo, hay diferentes formas de aprendizaje entre las que destacaremos el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo.

Aprendizaje supervisado

Caracterizado principalmente porque el aprendizaje se da de una forma controlada. Requiere un previo análisis de los datos que vamos a emplear para obtener ciertas características que nos permita realizar una clasificación o predicción. De tal manera que sabremos para cada entrada la salida correspondiente, pudiendo corregir la red cuando no se da la precisión deseada. En definitiva, podemos caracterizar este aprendizaje en que los datos se encuentran 'etiquetados' [13].

Dentro de este tipo de aprendizaje podemos destacar el aprendizaje por corrección de error, el cual se caracteriza porque va cambiando los pesos de las interconexiones cuando la salida del sistema no se corresponde con lo esperado, o el aprendizaje estocástico, el cual va variando de manera aleatoria los pesos de los enlaces evaluando si el cambio ha sido a mejor o a peor quedándonos solo con los cambios que mejoren los resultados.

Aprendizaje no supervisado

Este tipo de aprendizaje se diferencia en el anterior en que no disponemos de unos datos 'etiquetados'. Por ello debe ser la red la que extraiga los parámetros que considere oportunos para poder realizar una buena división en clases o una buena predicción.

Podemos destacar el aprendizaje hebbiano y el aprendizaje comparativo y competitivo [13].

Aprendizaje por refuerzo

Este puede considerarse un tipo de aprendizaje supervisado sin embargo en este caso tampoco tenemos una clara división de clases o un conocimiento concreto de la salida del sistema para las entradas disponibles.

La principal característica de este es la presencia de un supervisor que actúa como un ayudante para determinar si la salida obtenida se corresponde con lo esperado, empleándose esta información para ir determinando los pesos de las conexiones entre neuronas [13].

2.2.2.3.4 Redes convolucionales.

Se tratan de un tipo concreto de red neuronal que presenta grandes similitudes con la corteza visual primaria del cerebro. Estas son muy efectivas para tareas de visión artificial destacando su aplicación en clasificación de imágenes que será la que empleemos en el trabajo realizado [15].

Como indica su nombre se basan en la operación de convolución y son un tipo de red capaces de aprender relaciones entre una entrada y su correspondiente salida.

Centrándonos en su arquitectura podemos destacar 3 grandes capas. Las capas de convolución, las capas de pooling y las capas clasificadoras. Estas se estructuran como se ve a continuación:

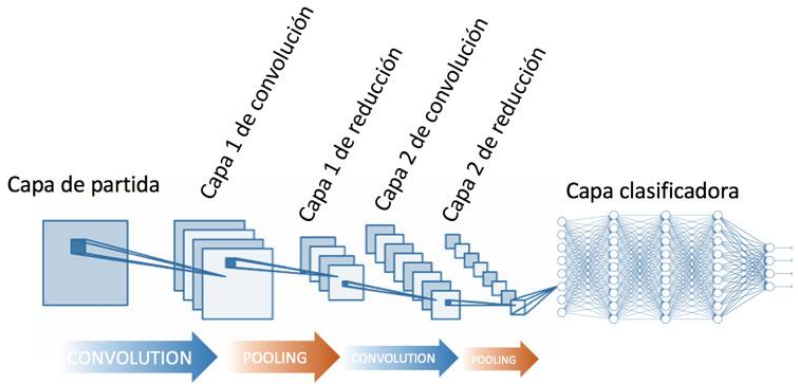

Figura 16 – Estructura de una red neuronal convolucional [15]

Como se puede apreciar las capas de convolución y pooling se van alternando entre ellas.

Capa de convolución

En esta capa se realizan diferentes tipos de operaciones, pero su función fundamental es la de actuar como filtro que dan lugar a un mapa de características nuevo como se explica en la siguiente imagen [15].

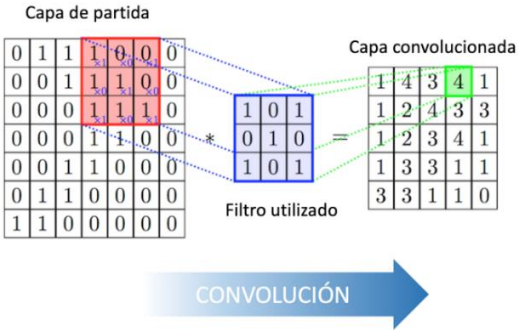

Figura 17 – Ejemplo de proceso de convolución [15]

En esta etapa de convolución cada píxel de la salida se corresponde a una combinación lineal de los pixeles de entrada.

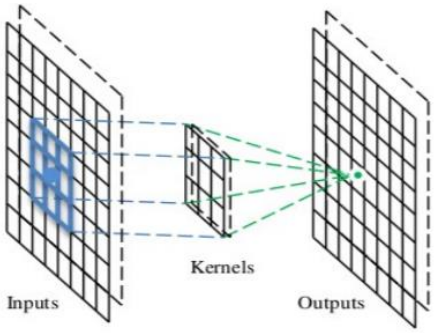

Figura 18 – Representación explicativa del proceso de convolución [16]

Cabe destacar que el filtro o kernel empleado también se va aprendiendo de modo que lo determinante es el tamaño del este para los resultados finales como podemos apreciar a continuación.

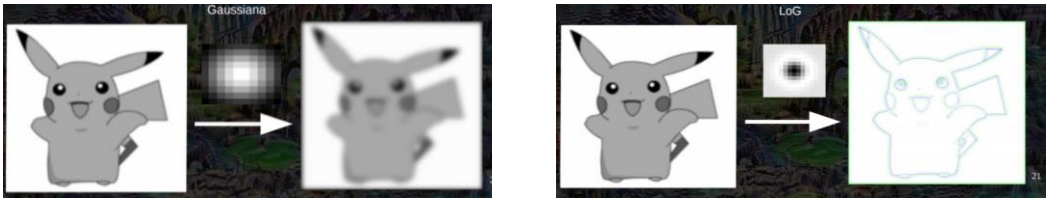

Figura 19 – Ejemplos sobre imágenes del proceso de convolución empleando dos kernel distintos [16]

Capa de pooling

Esta es la etapa de reducción, donde lo que pretendemos es quedarnos con aquellos parámetros y características más generales acorde a los datos [15].

El método empleado en la reducción puede ser la media de una región o el máximo de una región, entre muchos otros.

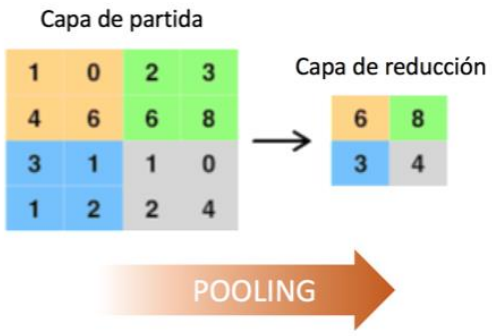

Figura 20 – Ejemplo del proceso de pooling [15]

Capa clasificadora

Esta es una capa totalmente interconectada cuya función es la que da su nombre, es decir, la de realizar una correcta clasificación.

La capa final dispondrá de tantas neuronas como clases posibles haya para los datos empleados.

2.2.2.3.5 Entornos de desarrollo.

Actualmente hay diversas librerías y entornos donde poder realizar diseño y desarrollo de estas redes artificiales. Entre ellos destacaremos los más empleados en la actualidad.

TensorFlow[2]

Es en la actualidad la librería más empleada. Desarrollada por Google, Tensorflow presenta numerosas ventajas. Entre ellas cabe destacar que su popularidad permite encontrar mayor número de contribuciones en GitHub. Del mismo modo gracias a la herramienta TensorBoard podemos visualizar de forma sencilla las operaciones que se van realizando, así como los datos empleados en las mismas lo cual permite localizar mejor los posibles errores.

Keras[3]

Se trata de otra librería de código abierto de redes neuronales, en Python. En este caso estamos ante una API de alto nivel que nos permite crear modelos con una sintaxis y estructura sencilla y rápida. Esto lo consigue gracias a que posee implementaciones de los bloques que conforman una red neuronal como son las diferentes capas, las funciones de activación, etc.

Caffe[4]

Es una arquitectura empleada para el desarrollo de modelos de aprendizaje profundo desarrollado en la universidad de California. Gracias a ella se pueden modelar CNNs, RCNNs orientadas, sobre todo, a la aplicación de estas en imágenes.

---

2 https://www.tensorflow.org/ 3 https://keras.io/ 4 https://caffe.berkeleyvision.org/

PyTorch[5]

Es un software dedicado al aprendizaje automático para el lenguaje de Python. Este se basa en la biblioteca Torch y nos ofrece como principales ventajas los tensores computacionales y las DNNs.

CNTK[6]

Es el último entorno que cabe destacar, desarrollado por Microsoft que afirma que se trata de un software mucho más eficiente que los vistos hasta el momento lo cual ha implicado que lleve a emplearse en áreas como el habla y el reconocimiento de imágenes.

2.2.2.3.6 Ventajas.

Sus principales ventajas son:

- Su capacidad de realizar un aprendizaje adaptativo, esto quiere decir que pueden ser entrenadas empleando una serie de datos de ejemplo, siendo capaces de realizar una distinción entre estos datos.
- Al igual que son capaces de realizar un aprendizaje adaptativo, pueden organizar la información con la que son entrenadas. Esta auto-organización es muy importante ya que permite llegar a un punto en el que estas redes se generalicen, pudiendo dar respuesta a situaciones desconocidas para ella, algo que resulta muy útil en aquellos casos en los que la información de entrada es un tanto difusa.
- Desarrollan una gran capacidad de tolerancia frente a errores como pueden ser datos incompletos o patrones con distorsión. Del mismo modo si una parte de esta red sufre un fallo y deja de funcionar, a pesar de que el resultado puede verse empeorado, la red es capaz de sobreponerse a ese error.
- Permiten llevar a cabo procesos a gran velocidad, debido a su implementación en paralelo.
- Y por último son fáciles de implementar en la tecnología existente.

2.3 Fusión de QbE-STD y redes neuronales profundas.

Actualmente es muy habitual que se junten diferentes técnicas con la finalidad de crear un nuevo sistema conjunto que una las ventajas que estas pueden presentar. El sistema QbE-STD ya fusiona dos maneras de afrontar el problema de la detección de palabras clave. A esta se le añadirá además una tercera técnica, muy popular en casos similares y que ha logrado buenos resultados, esta es la utilización de redes neuronales profundas.

No hay muchos ejemplos de desarrollos similares, sin embargo, en los últimos meses desde la realización de esta memoria han sido publicados 2 experimentos destacables. El primero de ellos el realizado por el Instituto de Investigación de Idiap, en Suiza, que aplicaba las CNNs (Redes Neuronales Convolucionales) [18] a un sistema QbE-STD sobre la base de datos de Spoken Web Search (SWS).

---

5 https://pytorch.org/ 6 https://docs.microsoft.com/en-us/cognitive-toolkit/

Y en segundo lugar nos encontramos con el Sistema Spoken Keyword Detection fusionado a un DTW-CNN desarrollado en conjunto por la Universidad John Hopkings de Baltimore y los Institutos Indios de Tecnología de Guwahati y de Dharwad, sobre la base de datos de TIMIT [19].

Como podemos apreciar, la fusión de estos subsistemas es algo cada vez más común en evaluaciones similares a la que nos encontramos. Es por ello que decidimos llevarlo a cabo.

Anteriormente, hemos mencionado también lo habitual que es aunar los sistemas de QbE y de STD. De este modo hay técnicas anteriormente mostradas que se emplean en conjunto en este escenario. Hablamos de la fusión de DTW y AKWS (Acoustic Keyword Spotting, sistemas basados en fonemas posteriores) [17]. A continuación, se observa un diagrama en el que se detalla una arquitectura como esta:

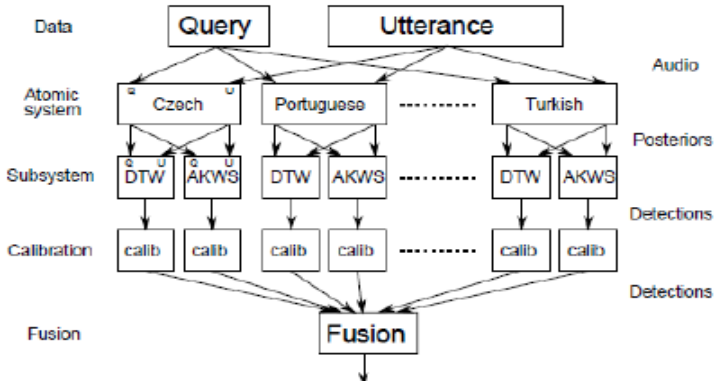

Figura 21 – Diagrama de bloques de un sistema QbE-STD con fusión DTW y AKWS [22]

Hemos comentado, lo efectivo y común que es la mezcla de distintos subsistemas, sin embargo, esto conlleva un proceso algo complejo de adaptación de las distintas partes con el fin de obtener unos óptimos resultados.

2.4 Precisión y factores a tener en cuenta.

Como se ha mencionado en la sección 2.1 la forma más común de evaluar sistemas como el desarrollado es ATWV. Del mismo modo otra forma es emplear el MTWV (Maximum Term Wighted Value) que representa el máximo ATWV obtenido por el sistema para diferentes valores de umbral empleados.

	ATWV	MTWV
Posterior + DTW [13] Fusión (MediaEval 2014, test data)	0,5841	0,6096
Posterior + DTW Relevance-feedback Fusión [20] (MediaEval 2012, test data)	0,7430	-

AKWS + DTW posterior **Fusión [16]** (MediaEval 2013, test data)	0,3751	0,3776
Posterior + DTW **Gaussian posterior + DTW** **Acoustic feat + DTW** **Fusión [18]** (Albayzin 2016, dev data)	0,2750	0,2800

Tabla 1 – Resultados de sistemas QbE-STD con fusión de subsistemas

Se puede ver en la tabla anterior como la fusión de sistemas ofrece resultados destacables. Otra forma de evaluación son las curvas DET (Detection Error Tradeoff) [3] que evalúan las prestaciones de los sistemas para distintas ratios de no detecciones y falsas alarmas.

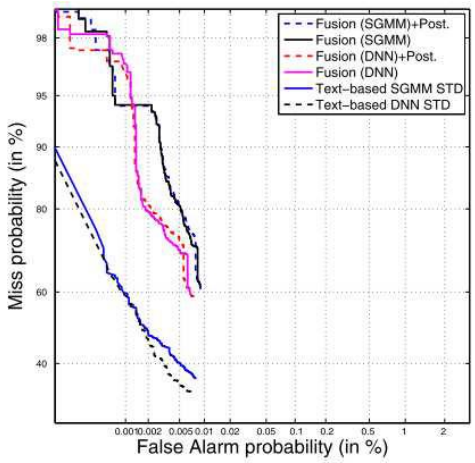

Figura 22 – Curvas DET [3]

Dentro de los factores a tener cuenta hay que destacar el **idioma**. Hasta el momento no hemos hecho incapié en esta característica, sin embargo, es un factor decisivo para el desarrollo del sistema y la utilización de unas técnicas u otras.

Debido a esto podemos realizar una clasificación entre los diferentes tipos de sistemas que nos encontramos. En primer lugar se encuenran los sistemas donde conocemos el idioma de queries y repositorio pudiendo realizar un entrenamiento específico. En segundo lugar estan los sistemas con pocos recursos donde se emplean técnicas que no requieren un gran conocimiento de los datos que se van a emplear. Y en último lugar los sistemas sin recursos o con cero recursos donde no empleamos ningún tipo de modelado relacionado con el idioma.

Capítulo 3

Diseño y desarrollo.

3.1 Elección del sistema a desarrollar.

Tras hacer un análisis de los posibles sistemas a llevar a cabo para a cometer el problema nos decantamos por el sistema QbE-STD basandonos en los posteriorgramas fonéticos como carácteristicas a tartar y empleando en este caso una red neuronal como bloque de decisión y para la búsqueda de las querys.

- Cabe destacar que los posteriorgramas fonéticos permitían el cálculo de matrices de coste con las que poder entrenar de una forma bastante sencilla a la red.

- Para la obtención de los posteriorgramas se empleó el reconocedor fonético de BUT ya que ofrecía buenos resultados pese a emplear en ocasiones queries en español.

- Se empleó el ya meniconado y explicado S-DTW debido a la gran diferencia de longitudes entre query y repositorio. Y también debido a que DTW proporciona buenos resultados en las condiciones en las que nos encontramos donde no sabemos el idioma de la query a buscar.

- Finalmente debido a los buenos resultados observados en trabajos similares nos decantamos por emplear las cada vez más impotantes redes neuronales como el bloque del sistema que nos permitiese localizar la query y decidir si se trataba de una ocurrencia positiva o negativa.

3.2 Entorno experimental.

El desarrollo del sistema se ha realizado en un entorno con sistema operativo Windows, alternando con Linux en función de la parte del proceso de desarrollo.

Los experimentos, como se mencionará, se han realizado en Google Colaboratory que cuenta con GPU pudiendo realizarse de forma remota con buenos resltados.

3.2.1 Base de datos.

La base de datos empleada se corresponde con los datos proporcionados para la evaluación Albayzing 2018 [24][25].

Esta evaluación cuenta con tres bases de datos distintas: MAVIR, COREMAH Y RTVE. De MAVIR se dispone de los conjuntos de *desarrollo*, *test* y *entrenamiento*. De

COREMAH solo se dispone de datos de *test* y por ultimo de RTVE disponemos de datos de *desarrollo*, *test* y *entrenamiento*.

Tanto MAVIR como COREMAH emplean codificación PCM, a 16kHz, con un único canal y 16 bits por muestra y RTVE utiliza AAC, stereo, 44.1 kHz y una tasa binaria variada.

Nos hemos centrado principalmente en la base de datos de MAVIR, empleando el conjunto *dev* durante toda la fase de desarrollo. Este conjunto esta formado por 2 ficheros de audio con una duración total de 1hora donde se buscarán las 102 queries de las que también se dispone.

Finalmente para realizar las evaluaciones y los experimentos se han empleado los datos de *test* de la base de datos de MAVIR.

3.2.2 Reconocedor fonético.

El reconocedor fónético empleado ha sido el desarrollado por la Universidad de Tecnología de Brno (BUT) [6] debido a la independencia que muestran para el idioma y por lo tanto los buenos resultados que se obtienen para queries en español, a pesar de no tener ningún modelado para este idioma en concreto.

Este reconocedor presenta opciones para los siguientes idiomas: inglés, checo, húngaro y ruso. Hemos empleado el reconocedor BUT en inglés para procesar nuestros datos ya que era el que mejores resultado obtenía.

Otra de las ventajas que ofrecía este reconocedor era su posibilidad de usarse tanto en un entorno Windows y Linux con gran facilidad. Debido a problemas encontrados al ejecutarlo esta parte la llevamos a cabo en Windows.

Centrándonos más en el funcionamiento de los reconocedores BUT hay que destacar que realiza la decodificación de los audios para después poder calcular los posteriorgramas correspondientes. Donde para cada unidad fonética se represenan tres estados: inicio, medio y final proporcionandose 3 probabilidades distintas para cada estado. Del mismo modo la salida tiene por tanto unas dimensiones de (T,3*U) siendo T la duración en frames del audio (cada frame equivale a 10ms) y U el número de unidades fonéticas (39 en nuestro caso)

3.2.3 HTK.

HTK se trata de un conjunto de herramientas con las que poder generar modelos ocultos de Markov, diseñada en un inicio para procesamiento de habla empleando estos modelos [7].

En el presente trabajo lo empleamos con el objetivo de poder transformar la salida del reconocedor fonético a texto plano. Obteniendo así ficheros que contengan los posteriorgramas que utilizaremos en nuestro sistema.

Haremos uso del programa HList para conseguir esta transformación. Este programa lo que hace es recibir a su entrada el fichero con las probabilidades para los 3 estados para todos los fonemas posibles y lo comprime. Para ello suma las probabilidades de los 3 estados consiguiendo asi un fichero en formato ASCII a su salida.

3.2.4 Google Colaboratory.

Este entorno fue empleado para el desarrollo del bloque de decision basado en redes neuronales.

Las principales razones de la elección de este entorno es la gran velocidad a la hora de procesar los datos que ofrecía. Esto unido a la facilidad de uso hizo que llevasemos a cabo los experimentos en él.

Cabe destacar que Google Colaboratory es gratuito y emplea Jupyter pudiendo ejecutar todo en la nube, además de que no necesita ningún tipo de configuración.

3.2.5 TensorFlow.

Como ya hemos explicado en la sección 2.2.2.3.5 TensorFlow es la librería más empleada en el diseño y desarrollo de modelos de aprendizaje automático y más concretamente de redes neuronales.

Esto unido a las ventajas que implica tener mayor información acerca de este sistema ha propiciado la elección del mismo.

3.2.6 Keras.

Es otro conjunto de librerías como ya se ha explicado en la sección 2.2.2.3.5. Esta permite elaborar CNNs de una forma bastante más sencilla y organizada. Esto unido a que se puede emplear conjuntamente con TensorFlow y que ambas bibliotecas son soportadas por Google Cloab han decantado la elección por estos entornos de desarrollo.

3.2.7 NIST.

Emplearemos la herramienta de cálculo de ATWV de NIST para hacer la evaluación con el fin de ver la eficacia de nuestro sistema. Como se ha mencionado en secciones anteriores (2.1 y 2.4) la evaluación ATWV es la más empleada en sistemas QbE-STD.

Para realizar el cálculo del ATWV y medir así la precision del sistema es necesario escribir un XML siguiendo una estructura concreta como la que se muestra a continuación:

```
<stdlist termlist_filename=
"/opt/Experimentos/MC/Evaluaciones/development/scoring/QbESTD/mcabello/results_EN_06-Aug-2018_154250.xml" indexing_time=
"1.000" language="spanish" index_size="1" system_id="fake">
<detected_termlist termid="DEV-0002" term_search_time="0.5" oov_term_count="1">
</detected_termlist>
<detected_termlist termid="DEV-0003" term_search_time="0.3" oov_term_count="1">
<term file="mavir03" channel="1" tbeg="1195.690" dur="0.280" score="-0.03393" decision="YES"/>
</detected_termlist>
<detected_termlist termid="DEV-0007" term_search_time="0.5" oov_term_count="1">
<term file="mavir03" channel="1" tbeg="352.050" dur="0.150" score="-0.08310" decision="YES"/>
<term file="mavir03" channel="1" tbeg="1576.960" dur="0.420" score="-0.09187" decision="YES"/>
<term file="mavir03" channel="1" tbeg="1854.330" dur="0.400" score="-0.09371" decision="YES"/>
</detected_termlist>
</stdlist>
```

Figura 23 – Ejemplo de documento XML para usar en la evaluación

Cabe destacar como partes más importantes los siguientes conceptos del XML anterior:

- *termed* que indica la query detectada.
- *Term_search_time* que expresa el tiempo que dura la query en segundos.
- *tbeg* el instante de tiempo donde comienza la ocurrencia.
- *file* el repositorio donde se encontró la query.
- *dur* que indica la duración de la ocurrencia.

3.3 Desarrollo del sistema.

El desarrollo del sistema QbE-STD se ha dividido en diferentes partes para poder lograr los objetivos parciales mencionados en la sección 1.2. Finalmente, el resultado es un sistema desarrollado en Matlab y Google Colaboratory que trata de maximizar la automatización del proceso a realizar para lo que se ha necesitado:

- Lista de las queries de *test* y *train* que se quieren buscar.

- Lista de repositorios donde realizar la búsqueda de las queries.

- Es necesario saber el idioma con el que queremos realizar el experimento, para así poder emplearlo en el reconocedor fonético. Estos idiomas pueden ser checo, húngaro, ruso o inglés.

- Parámetros necesarios para el algoritmo DTW y poder generar las matrices que procesaremos y emplearemos en el bloque siguiente.

- Conjunto de imágenes de casos positivos y negativos para entrenar y testear la red de decisión.

3.3.1 Obtención de posteriorgramas.

Para la obtención de posteriorgramas es necesario llevar a cabo un proceso concreto. En primer lugar, debemos hacer uso del reconocedor fonético BUT explicado en la sección 3.2.2. Como hemos mencionado empleando la función *phnrec* podemos obtener una matriz de probabilidades a posteriori cuyo tamaño es (T, 3*U) siendo T el número de frames de 10ms cada uno y U el número de fonemas disponibles para el idioma seleccionado. El 3 se debe a cada uno de los estados del fonema.

Después de este es necesario hacer uso de la función HList que tenemos en el conjunto HTK. Gracias a esto podremos además realizar una conversión del formato, pasando de *.post* a *.txt* y consiguiendo una matriz de la siguiente forma:

$$P = \begin{pmatrix} p_{1,1,t_1} & p_{1,2,t_1} & p_{1,3,t_1} & \cdots & p_{U,1,t_1} & p_{U,2,t_1} & p_{U,3,t_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p_{1,1,T} & p_{1,2,T} & p_{1,3,T} & \cdots & p_{U,1,T} & p_{U,2,T} & p_{U,3,T} \end{pmatrix}$$

Como podemos observar tenemos una matriz donde para cada unidad fonética tenemos 3 probabilidades correspondientes a cada estado. Con el fin de reducir la dimensión de

esta matriz y tener una sola probabilidad para cada fonema, procedemos sumando, para cada fonema, las probabilidades correspondientes a sus 3 estados obteniendo como resultado una matriz de tamaño (T, U).

Las dimensiones de las matrices correspondientes a los posteriorgramas son 120 en el eje X debido a las 40 unidades fonéticas que tiene la lengua inglesa y el eje Y varía en función de la duración en frames de la locución, como ya hemos explicado anteriormente.

NOTA: Para realizar esta parte se elaboraron los scripts correspondientes y así poder hacer de forma automatizada la ejecución de los comandos para todas las queries y repositorios disponibles.

3.3.2 Tratamiento y adaptación de la base de datos.

Con la finalidad de poder elaborar un método con el que poder realizar una comparación entre query y repositorio de búsqueda, partimos de una implementación explicada en la sección 2.2.2.1 S-DTW.

Gracias a este bloque podemos obtener la matriz de coste que nos permite medir la similitud entre los audios siguiendo un esquema muy sencillo.

Figura 24 – Diagrama de bloques de cálculo de matriz de coste

A pesar de la simplicidad del esquema anterior, este bloque requiere una parte de cálculo de mayor complejidad.

Para realizar el cálculo de esta matriz establecemos como medida de comparación de distancia local el coeficiente de correlación de Pearson, *r* debido a los buenos resultados que ofrece.

Cada valor de la matriz será mayor si la secuencia de audio y subsecuencia del repositorio son parecidas (más correladas) y más bajo en caso de no presentar similitud (menos correladas). Las dimensiones de la matriz de coste serán (Frames Query, Frames Repositorio) y el cálculo se realiza siguiendo la siguiente fórmula [5]:

$$r(x_n, y_m) = \frac{U(x_n \cdot y_m) - \|x_n\|\|y_m\|}{\sqrt{(U\|x_n^2\| - \|x_n\|^2)(U\|y_m^2\| - \|y_m\|^2)}}$$

Como hemos dicho anteriormente cuando las muestras están alineadas presentaran un alto valor de correlación *r*, esto se traduce en que el coste será más bajo cuando el valor de *r* sea más alto. Es por ello que realizamos la siguiente transformación:

- Si *r* es igual a -1 → el coste *c* será igual a 1
- Si *r* es igual a 0 → el coste *c* será igual a 0.5
- Si *r* es igual a 1 → el coste *c* será igual a 0

A esto se le añade una modificación con el fin de ser más discriminativos entre muestras alineadas y no alineadas. Este cambio consiste en dar un coste máximo de 1 a valores de *r* menores o iguales de 0.

Una vez calculada la matriz de coste, esta tiene una forma distinta y diferenciable para casos positivos (diagonal de camino óptimo con coste cero) y casos negativos, observándose ambos ejemplos a continuación.

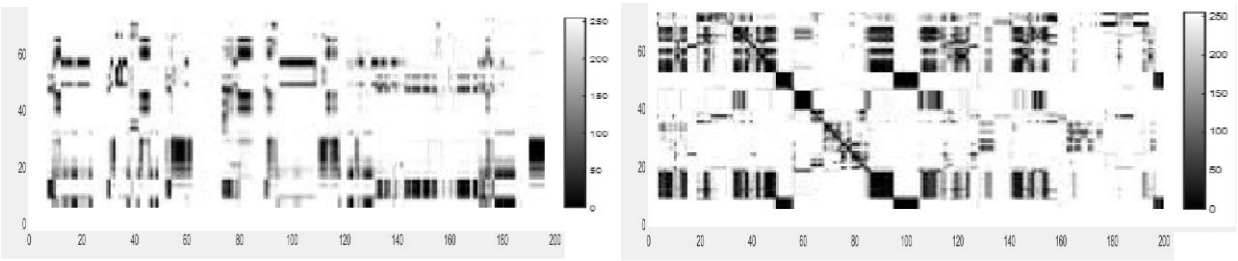

Figura 25 – Matriz de coste de una ocurrencia positiva (derecha) y de una negativa (izquierda)

Una vez calculada esta matriz es posible llegar a la matriz de coste acumulado, esta también ha sido tenida en cuenta en el desarrollo para realizar una comparación entre los datos obtenidos con una y otra representación.

Esta matriz de coste acumulado ha sido obtenida realizando el siguiente cálculo:

$$D(n,1) = \sum_{k=1}^{n} c(x_k, y_1)\ para\ n \in [1:N]$$

$$D(1,m) = c(x_1, y_m)\ para\ m \in [1:M]$$

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

$$para\ el\ resto\ de\ los\ casos$$

A continuación, podemos ver las matrices de coste y de coste acumulado de un caso de ocurrencia positiva:

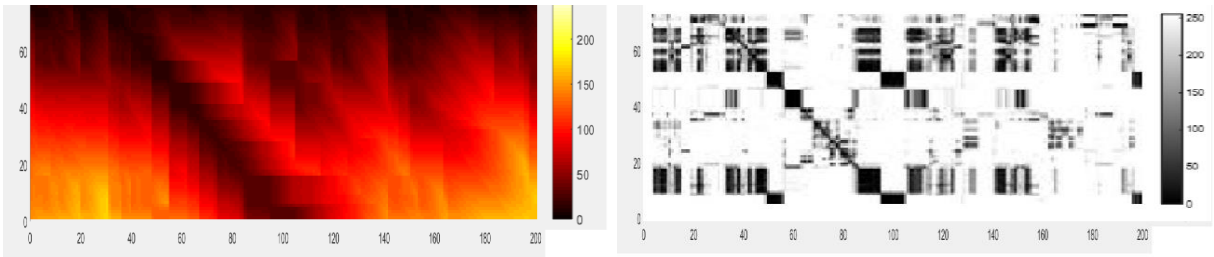

Figura 26 – Matriz de coste (derecha) y de coste acumulado (iaquierda)

Se aprecia de forma clara la zona de máximo alineamiento donde query y subsecuencia son más parecidas. Al ser apreciable de forma visual para el hombre hace que sea posible una fácil detección por medio de redes neuronales artificiales.

Hasta ahora hemos visto la parte matemática, sin embargo, como hemos mencionado anteriormente, de estos cálculos obtenemos una matriz de coste (o coste acumulado) cuyas dimensiones son (Frames Query, Frames repositorio de búsqueda). Esto hace que

tengamos matrices de un tamaño desproporcionado entre filas y columnas siendo de por ejemplo 74 filas y 344282 columnas.

Si pasásemos a la red neuronal una matriz de este tamaño el aprendizaje sería nulo, es por ello que necesitamos hacer una segmentación de la misma. Tras realizar el análisis oportuno de todos los datos observamos que la duración de las queries no supera en su mayoría el 1 segundo quedando por debajo de esta cifra el 98% de las muestras de desarrollo y test.

Por tanto, los valores que hemos establecido para realizar el 'enventanado' han sido de 2 segundos para el ancho de la ventana y de 1 segundo para el desplazamiento. Esto traducido a frames (pixeles o muestras en la imagen) equivale a 200 y 100 respectivamente, ya que como sabemos 1 frame = 10ms.

Tomamos un valor de ventana casi del doble o más de las queries para que estas entren de forma completa en una sola ventana. El desplazamiento es igual, aproximadamente, a la máxima duración de las queries, esto lo decidimos por dos motivos. Primero, con este desplazamiento logramos que la query siempre este dentro completamente de una ventana y en el caso de tenerla al final de una de ellas poder tener de una misma ocurrencia dos casos positivos para posteriormente entrenar a la red. Con esto además acostumbramos a la red a que las ocurrencias no tienen que darse siempre al principio o al final de la imagen, sino que pueden darse en cualquier lugar de la ventana.

Esto se explica mejor con el siguiente ejemplo en el que tenemos una query en las últimas muestras de la ventana 6200-6400. Al desplazarnos a la ventana 6300-6500 pasamos a tener la misma ocurrencia en otra posición, relativa a la ventana de análisis en cada instante, obteniendo así dos imágenes como hemos dicho anteriormente.

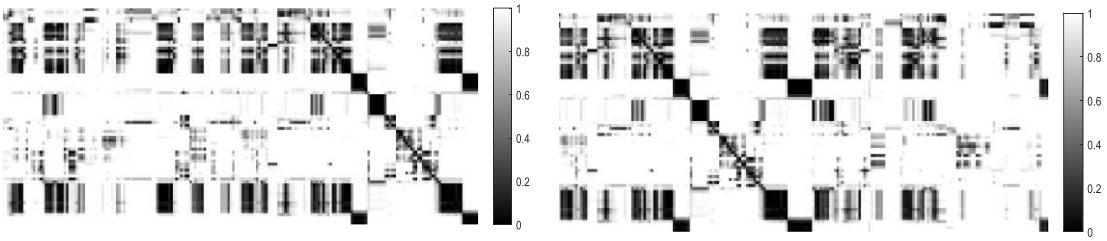

Figura 27 – Matriz de coste de la misma ocurrencia, tomando ventanas distintas

3.3.3 Red Neuronal de decisión.

Esta es la parte más novedosa de nuestro sistema QbE-STD. Las redes neuronales artificiales se emplean en la actualidad de una forma muy común en el reconocimiento de imágenes y podemos aprovecharlo.

Tras realizar una exhaustiva documentación a cerca de las redes que mejor podrían funcionar en casos similares al que nos encontramos, llegamos a distintos ejemplos que nos servirían de base para el diseño de nuestra red.

Partimos de una red convolucional entrenada para la base de datos de MINST [23], esta se emplea para la clasificación de imágenes en escala de grises, es por ello que decidimos escogerla. Esta red está formada en un inicio por:

CAPA	Descripción
Entrada	100x200x1
Conv	Kernel 3x3 Activación ‘relu’
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Dropout	Abandono del 25%
Flatten	-
Dense	Activación ‘relu’
Dropout	Abandono del 50%
Dense	Activación ‘softmax’
Salida	Clase = {0,1}

Tabla 2 – Esquema de red neuronal diseñada

La función de cada una de las capas es sencilla. Como ya hemos mencionado anteriormente en la sección 2.2.2.3.4 las capas de convolución sirven para realizar el filtrado en este caso empleando un filtro 3x3 y la capa de polling para finalmente hacer la reducción y así quedarnos con las características más generales. Con las capas de Dropout o de abandono se busca desconectar una parte de las neuronas y comprobar el funcionamiento para así equilibrar los pesos de los enlaces y que no siempre una neurona lleve a cabo una sola función. Finalmente, las capas Dense y Flatten sirven como últimas para obtener la salida del sistema.

A estas capas anteriormente estructuradas añadimos más operaciones de convolución y pooling, concretamente una capa de cada una. Esto debido a que las características observadas por los datos así nos lo requerían.

Además de esta red mencionada, continuamos con la indagación de posibles problemáticas similares. Es así como llegamos al caso de ‘CNN based Query by Example Spoken Term Detection’ [18] trabajo ya mencionado anteriormente, realizado por el Instituto de Investigación de Idiap. En este caso se tomaba como red de partida la VGG la cual había dado grandes resultados en reconocimiento de imágenes.

Como sabemos esta es una buena noticia ya que estamos tratando nuestras matrices de coste como imágenes. En este caso solo con un único canal RGB. Finalmente, la arquitectura presentada por la red utilizada tiene cuatro conjuntos de dos capas de convolución y una capa de agrupación máxima (MaxPolling) y por último dos capas totalmente conectadas con un soft-max en la parte superior. A continuación, se puede observar la estructura final de la red.

CAPA	Descripción
Entrada	100x200x1
Conv	Kernel 3x3 Activación ‘relu’
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2

Conv	Kernel 3x3 Activación ‘relu’
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Conv	Kernel 3x3 Activación ‘relu’
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Conv	Kernel 3x3 Activación ‘relu’
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Capa totalmente conectada	-
Capa totalmente conectada	-
Salida	Activación ‘softmax’ Clase = {0,1}

Tabla 3 – Esquema de red neuronal diseñada (2)

La función de activación empleada en las capas convolucionales ha sido ‘relu’. El número de canales y la tasa de abandono se otimizaron a 30, y 0,2 respectivamente con un conjunto de desarrollo.

Finalmente, la arquitectura presenta ocho capas convolucionales en total. Esto nos llevo a pensar que posiblemente no necesitásemos tantas y realizamos un diseño similar donde fuimos aumentando el número de capas de forma progresiva hasta finalmente llegar a la red con la que más experimentos realizamos debido a los buenos resultados que ofrecía.

La estructura de la red final es la siguiente:

CAPA	Descripción
Entrada + Conv	100x200x1 Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Conv	Kernel 3x3 Activación ‘relu’
Maxpool	Kernel 2x2
Flatten	Activación ‘relu’
Dense	-
Dense + Salida	Activación ‘sigmoid’

Tabla 4 – Esquema de red neuronal definitiva

Consta de 3 capas convolucionales, que fueron suficientes para la clasificación de nuestras matrices de coste.

Entrenamiento del modelo.

Esta es la segunda parte que necesitó un diseño más exhaustivo. En un principio contábamos aproximadamente con 300 casos positivos y 300 casos negativos. Decidimos balancear este número para no mal acostumbrar a la red a solo trabajar con un tipo de datos y que vea los dos por igual. Esto es algo irreal ya que las ocurrencias son muy inferiores respecto a las no ocurrencias, sin embargo, es algo necesario como hemos mencionado anteriormente.

Después de comprobar que el número de datos de entrenamiento era algo bajo, conseguimos crear varias ocurrencias positivas y sus correspondientes negativas como se explicará más adelante. De esta manera contamos con 1740 casos positivos y sus respectivos casos negativos.

Una vez definidos los datos que emplearíamos en la fase de entrenamiento y validación no realizamos un entrenamiento estándar. Esto se debió a que nos vimos en la obligación de crear una función que actuase como nuestro generador de datos. Esta recibe el nombre de *my_generator* y su función era la siguiente.

- Debía elaborar una lista inicial con los datos de entrenamiento. Y aleatorizarla para que se mezclen los datos positivos con los negativos.
- Una vez conseguíamos esto debíamos mandar a nuestra red solo una pequeña parte de los datos totales, de tal manera que en cada paso la red viese 40 datos aleatorizados entre positivos y negativos y en cada época pudiese ver todos los datos disponibles.
- Del mismo modo cada vez que terminaba una época era necesario realizar una aleatorización del orden de los datos obteniendo una nueva lista y enviando los datos en un orden cada vez distinto de tal manera que para la red se viesen como datos nuevos.

Finalmente, el número de épocas se determinó de forma experimental siendo con 25 con la que mejores resultados se obtenían, aunque esto podría implicar más tiempo de ejecución. Un dato destacable de la etapa de entrenamiento es que debido a la escasez de los datos de desarrollo se empearon los de test para entrenar la red. De este conjunto total se empleó el 20% como datos de testeo entrenando finalmente con 2794 datos y validando con 699.

3.3.4 Problemas encontrados.

En función fue avanzando el diseño y desarrollo del sistema nos enfrentamos a distintos problemas. Entre ellos cabe destacar 3.

Datos sintéticos.

Este problema surge debido a que en un inicio el cáluclo de las matrices de coste no se estaba realizando correctamente. De tal manera que había casos positivos que se observaban con gran claridad y del mismo modo había otros practicamente indetectables. A continuación, se puden apreciar 2 casos positivos, uno fácilmente detectable y otro que no se distingue de un caso negativo.

Figura 28 – Caso positivo fácilmente diferenciable (izquierda) y caso positivo no diferenciable (derecha)

Es por ello que pensamos que si era complicado para nosotros hacer la clasificación podría ser casi imposible para la red poder generar un aprendizaje apartir de estos datos.

Por lo tanto, realizamos las comprobaciones para el cálculo de las matrices de coste donde se comparaban repositorio de búsqueda con segmentos del propio repositorio. Obteniendose resultados bastante mejores de ocurrencias positivas.

El siguiente paso fue realizar un nuevo generador de datos, este lo hicimos sobre matlab. Y empleamos los ficheros de ocurrencias proporcionados y los ground-truth de los datos de test para elaborarlos. Es asi como conseguimos llegar a un número más grande de datos de entrenamiento pasando de tener 600 a algo más de 2000.

Ventanas de análisis.

Como consecuencia de lo anterior surgió otro problema, sin embargo, este tenía fácil solución. Al elaborar los datos sintéticos lo que provocamos fue que la mayoría de ocurrencias se diese al comienzo de la ventana de análisis. Esto hacía que nuestro sistema tuviese un aprendizaje limitado y que en casos donde el alineamiento es máximo en zonas más tardías de la ventana no se detectasen con éxito.

Para resolver este problema procedimos de la siguiente manera. En aquellos casos donde las ocurrencias se daban muy próximas al inicio de la ventana, tomábamos como dato tanto esa ventana como la anterior. Debido al desplazamiento de 100 frames conseguiamos tener la ocurrencia en un punto de la ventana más atrasado. Esto es similar a lo explicado anteriormente donde de 1 caso positivo obteníamos 2 imágenes de entrenamiento para la red.

Matriz de coste vs. Matriz de coste acumulado.

Esta fue otra de las principales preocupaciones. Debido a que debíamos elegir la opción que permitiese a la red realizar un trabajo más eficiente y óptimo. Es por ello que nos decantamos por la matriz de coste donde pensamos que se veían de una forma más clara la distinción entre casos positivos y negativos.

Capítulo 4

Pruebas y resultados.

A continuación, podremos tener ua vision de los experimentos realizados y cómo fuimos cambiando distintas partes para elaborar el mejor sistema posible.

4.1 Variación de los datos empleados.

Esta parte se puede dividir en 2 secciones correspondientes a cada uno de los cambios importantes que sufrieron nuestros datos durante la experimentación y que llevaron a resultados distintos.

Empleando los datos originales.

Este fue el primero de los experimentos llevados a cabo donde utilizamos los datos obtenidos de las queries originales. Contábamos exactamente con un total 329 datos positivos y negativos. En este caso obtuvimos unos valores de *accuracy* y *loss* sobre los datos de entrenamiento como los que se observan a continuación.

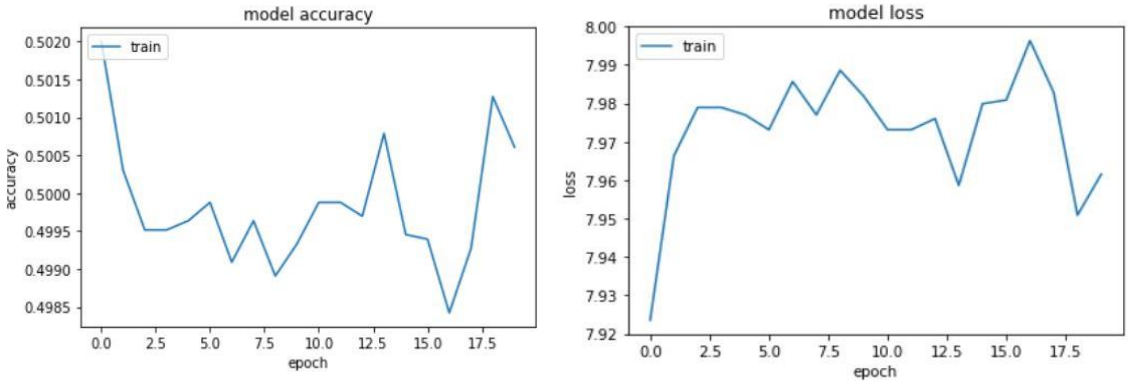

Figura 29 – Valores de accuracy y loss para el experiment empleando los datos originales

Como podemos observar los resultados obtenidos nos llevaban a pensar en que había algún fallo con los datos que estábamos utilizando. Es así como nos dimos cuenta del problema que presentaban en el cálculo de las matrices de coste. Y nos llevó a la elaboración de los, ya mencionados en la sección anterior, datos sintéticos.

Empleando datos sintéticos.

En este caso continuamos con la realización de experimentos con los nuevos datos generados de forma artificial. Contamos con un total de 3493 datos. Con esto obtuvimos una mejora algo significativa en los valores anteriores de *accuracy* y *loss* como se muestra a continuación sobre los datos de validación consiguiendo un máximo en la *acuraccy* de 0.8354.

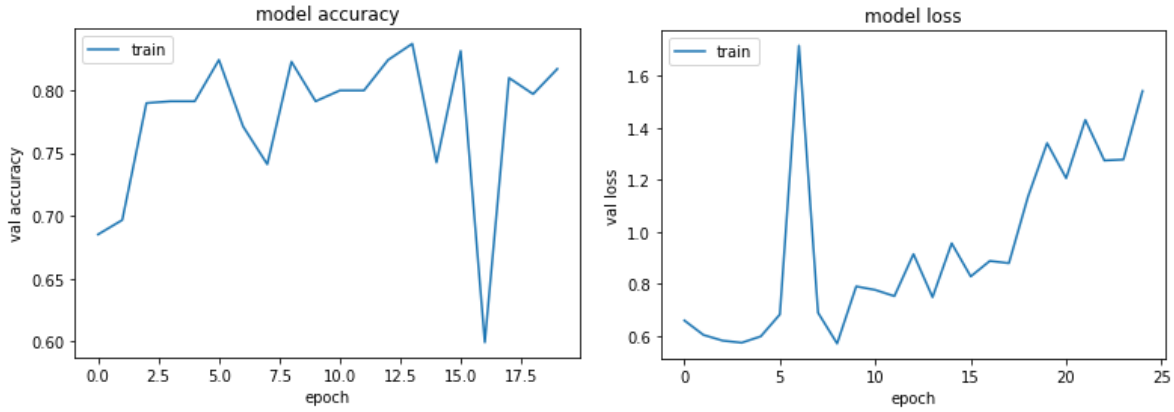

Figura 30 – Resultados de accuracy y loss empleando datos sintéticos de validación

NOTA: También se realizaron experimentos cambiando el modo de entrenaiento, sin embargo, los resultados no variaron en exceso.

4.2 Variación de los parámetros de la red.

Respecto a los parámetros más detacables de la red, solo realizamos cambios en uno de ellos que fueron las épocas del entrenamiento. También se pensó en cambiar las funciones de activación pero la función 'relu' tiene un gran respaldo en el reconocimiento de imágenes.

Los cambios que fue sufriendo el parámetro de las épocas cambiaba el número de veces que nuestra red veía nuestros datos. Probamos con 15, 20 y 25 épocas obteniendose los siguientes resultados.

15 Épocas:

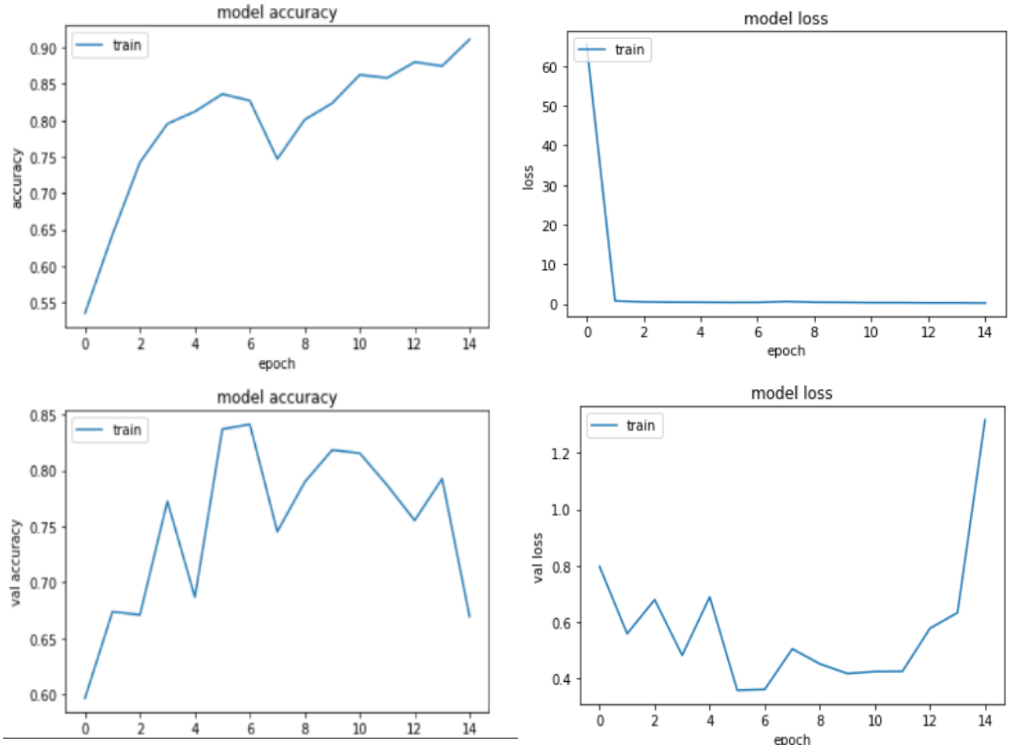

Figura 31 – Valores obtenidos para accuracy y loss para los datos de entrenamiento (arriba) y para los datos de validación (abajo)

En este caso es sorprendente como fijando el número de épocas a 15 los resultados obtenidos por la red mejoran considerablemente respecto a las 10 que se probaron en un inicio. Teniendo una exactitud máxima sobre los datos de validación del 0.8432.

Datos	Accuracy media	Loss media
Entrenamiento (MAVIR)	0.7935	0.7164
Validación (MAVIR)	0.7473	0.2958

Tabla 5 – Resultados de accuracy y loss de la red empleando 15 épocas

20 Épocas:

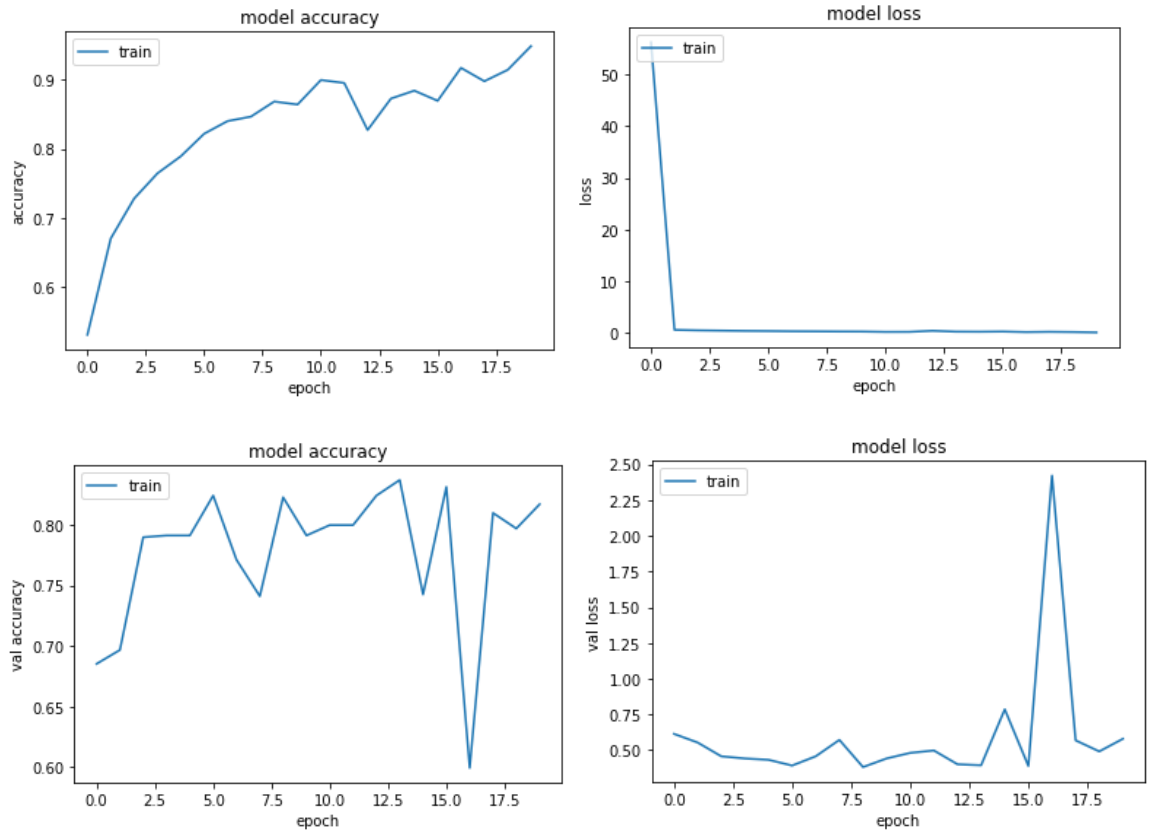

Figura 32 – Valores obtenidos para accuracy y loss para los datos de entrenamiento (arriba) y para los datos de validación (abajo) (2)

Como cabía esperar al ver los resultados obtenidos en el caso anterior, a partir de ahora la precisión con la que realiza la clasificación nuestra red es mayor según aumentamos el número de épocas. En este caso sin embargo podemos observer un pico muy marcado en la época 16 donde la red presenta dificultades, esto puede deberse a la sobrecarga de alguna de las zonas, ya que este problema no se volvió a repetir.

Datos	Accuracy media	Loss media
Entrenamiento (MAVIR)	0.8800	0.7114
Validación (MAVIR)	0.7780	0.7100

Tabla 6 – Resultados de accuracy y loss de la red empleando 20 épocas

25 Épocas:

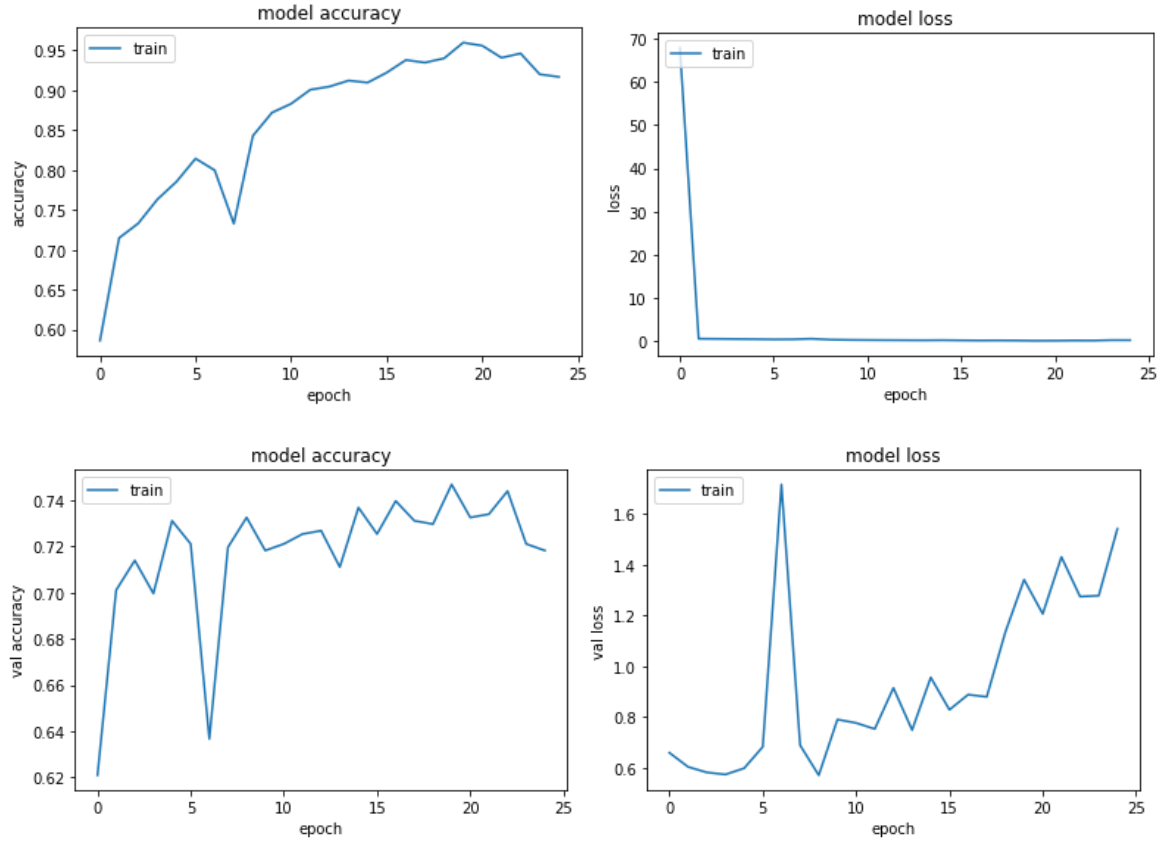

Figura 33 – Valores obtenidos para accuracy y loss para los datos de entrenamiento (arriba) y para los datos de validación (abajo) (3)

Como ya se ha dicho los resltados mejoraban aumentando la carga computacional, sin embargo, teníamos que establecer un límite ya que no podríamos entrenar nuestro modelo eternamente. De esta manera llegamos a las 25 épocas que son las que mejores resultados nos ofrecieron. A partir de aquí se puede aumentar el número de épocas, sin embargo, la mejora es mínima, siendo contraproducente cargar de un trabajo excesivo a nuestra red.

Datos	Accuracy media	Loss media
Entrenamiento (MAVIR)	0.9100	0.6350
Validación (MAVIR)	0.7213	0.8120

Tabla 7 – Resultados de accuracy y loss de la red empleando 25 épocas

NOTA: Todos esos datos se corresponden a los experimentos realizados con la matriz de coste (no acumulado).

Capítulo 5

Conclusiones y trabajos futuros.

5.1 Conclusiones.

En la realización de este Trabajo Fin de Grado se ha tratado de implementar un nuevo sistema QbE-STD que incorporase una red neuronal capaz de poder localizar las ocurrencias.

Hemos empleado numerosos conceptos desarrollados durante la carrera, como son los vectores de características en nuestro caso posteriorgramas, una variante de DTW para poder realizar el cálculo de las correspondientes matrices de coste y de coste acumulado y poder medir la similitud entre ambas y por último el aprendizaje automático y más concretamente las redes neuronales como método de búsqueda y reconocimiento.

Se ha podido ver cuales son las técnicas que nos pueden ofrecer mejores resultados en un caso de sistema con recursos escasos. Llevándonos a emplear el idioma inglés para nuestros datos en castellano en el reconocedor fonético BUT, ya que ofrecía los mejores reusltados.

Cabe destacar que la aportación propia para este sistema fue la implementación de la red neuronal, para sustituir la búsqueda de camino óptimo empleada anteriormente. Esto debido a las ventajas que proporcionaban y a su buen comportamiento en problemáticas similares.

Nos planteamos el objetivo de mejorar los resultados obtenidos en el sistema anterior, sin embargo, debido a diversos problemas como se han mencionado anteriormente esto no ha podido llevarse acabo de forma objetiva ya que no fue posible hacer una evaluación NIST correcta. Aún así se seguirá mejorando la implementación realizada pudiendo ser la continuación de este proyecto un posible Trabajo Fin de Máster donde trabajaremos en dicha implementación. Pudiendo evaluarla y compararla con sistemas similares.

5.2 Trabajos futuros.

Como ya hemos mencionado, esta implementación deja unas sensaciones de que tiene un amplio margen de mejora. Pudiendo ser material de studio e investigación para un futuro Trabajo Fin de Máster. Del mismo modo se plantea como sistema para poder presentarse a futuras evaluaciones.

Se pueden realizar optimizaciones en distintos aspectos como el cálculo de las matrices de similitud, el diseño de la red de decisión y sobre todo en los datos empleados. Al emplear en nuestro sistema queries en castellano con un reconocedor en inglés nos hace pensar que podríamos mejorar los resultados utilizando bases de datos en inglés.

Bibliografía

[1] Ramakrishnan, Raghu; Gehrke, Johannes, "6. QBE" (PDF), Database Management Systems (3rd ed.), Wisc.

[2] D. Torre-Toledano, J. Tejedor, 'Search-on-Speech: Recuperación de información en repositorios de audio', Noviembre de 2014

[3] J. Tejedor, D. Torre-Toledano, P. Lopez-Otero, L. Docio-Fernandez, and C. Garcia-Mateo, "Comparison of ALBAYZIN Query-by-example Spoken Term Detection 2012 and 2014 Evaluations," EURASIP J. Audio, Speech, Music Process., pp. 1–19, 2016.

[4] M. Müller, *Information retrieval for music and motion*. 2007.

[5] M. Cernak, A. Asaei, and H. Bourlard, "On structured sparsity of phonological posteriors for linguistic parsing," *Speech Commun.*, vol. 84, pp. 36–45, 2016.

[6] P. Schwarz, "Phoneme recognition based on long temporal context," *PhD Thesis, Brno Univ. Technol.*, 2008.

[7] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Lui, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.4)*, Cambridge, UK, 2006.

[8] Namrata Dave, "Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition", en International journal for advance research in engineering and technology, volumen 1, número 6, Julio de 2013.

[9] Lifa Sun, Kun Li, Hao Wang, Shiyin Kang and Helen Meng, 'Phonetic posteriorgrams for many-to-one voice conversion without parallel data training', 2016.

[10] Thomas Pratzlich, Jonathan Driedger and Meinard Müller, 'Memory-restricted multiscale dynamic time warping', Marzo 2016.

[11] Stan Salvador and Philip Chan, 'FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space', 2004.

[12] Doroteo Torre Toledano y Joaquín González Rodríguez, 'HMMs', Septiembre 2010

[13] Damián Jorge Matich, 'Redes Neuronales: Conceptos Básicos y Aplicaciones', Marzo 2001.

[14] Pedro Larranaga, ˜ Inaki ˜ Inza, Abdelmalik Moujahid, 'Tema 8. Redes Neuronales', Departamento de Ciencias de la Computaci´on e Inteligencia Artificial Universidad del Pa´ıs Vasco–Euskal Herriko Unibertsitatea.

[15] Diego Calvo, 'Red Neuronal Convolucional CNN', Julio 2017. (available on: http://www.diegocalvo.es/red-neuronal-convolucional/ Accessed: Junio 2019)

[16] Patricio Loncomilla, ‘Deep learning: Redes Convolucionales’. Instituto Nacional de Astrofísica Óptica y Electrónica.

[17] Igor Szöke, Miroslav Skácel, and Lukáš Burget, ‘BUT QUESST 2014 System Description’, 2014.

[18] Dhananjay Ram, Lesly Miculicich, Herv´e Bourlard, ‘CNN based Query by Example Spoken Term Detection’, 2018.

[19] Ravi Shankar, C.M. Vikram, and S.R.M. Prasanna, ‘Spoken Keyword Detection using joint DTW-CNN’, Septiembre 2018.

[20] D. Torre-Toledano, ‘Audio Mining en Big Data: Speech-To-Text and Searchon-Speech.’, Marzo 2019.

[21] Javier Ortega García, ‘Tema 3: Sistema Auditivo, Sensación Sonora y Parametrización Perceptual’, Grupo de Reconocimiento Biométrico – ATVS, Escuela Politécnica Superior, Universidad Autónoma de Madrid.

[22] I. Szöke, L. Burget, F. Grézl, J. “Honza” Černocký, and L. Ondel, “CALIBRATION AND FUSION OF QUERY-BY-EXAMPLE SYSTEMS - BUT SWS 2013, BUT Speech @ FIT, Brno University of Technology, Czech Republic,” pp. 7899–7903, 2014.

[23] MNIST database From Wikipedia, the free enciclopedia, Junio 2019.

[24] J. Tejedor and D. T. Toledano, “The ALBAYZIN 2016 Search on Speech Evaluation Plan,” pp. 1–11, 2016 (available on: https://iberspeech2016.inescid.pt/index.php/albayzin-evaluation/, Accessed: 6 Sept. 2018).

[25] J. Tejedor and D. T. Toledano, “The ALBAYZIN 2018 Search on Speech Evaluation Plan,” pp. 1–13, 2018 (available on: http://iberspeech2018.talp.cat/index.php/albayzin-evaluation-challenges/search-onspeech-evaluation/, Accessed: 6 Sept. 2018).

[26] Javier Ortega García, ‘DTW. Alineamiento Temporal Dinámico (DTW)’, Grupo de Reconocimiento Biométrico – ATVS, Escuela Politécnica Superior, Universidad Autónoma de Madrid.

[27] M. Müller, *Information retrieval for music and motion*. 2007.

[28] Maria Cabello, D. Torre-Toledano, J. Tejedor, ‘AUDIAS-CEU: A Language-Independent Approach for the Query-by-Example Spoken Term Detection task of the Search on Speech ALBAYZIN 2018 Evaluation’, Diciembre 2018.

[29] L. Muda, M. Begam, and I. Elamvazuthi, “Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques,” vol. 2, no. 3, pp. 138–143, 2010.

[30] I. Szöke, L. Burget, F. Grézl, J. “Honza” Černocký, and L. Ondel, “CALIBRATION AND FUSION OF QUERY-BY-EXAMPLE SYSTEMS – BUT SWS 2013, BUT Speech @ FIT, Brno University of Technology, Czech Republic,” pp. 7899–7903, 2014.

[31] Minaya Villasana, ‘Introducción a las redes neuronales (neurales) CO-6612’, Enero 2013. (available on: https://es.slideshare.net/HALCONPEREGRINO2/introduccion-a-las-redes-neuronales, Accessed: 14 Jun. 2019).

[32] Dmitriy Parfenovich, ‘Redes Neuronales: de la teoría a la práctica’, Mayo 2014. (available on: https://www.mql5.com/es/articles/497, Accessed: 14 Jun. 2019).