UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

EXTRACCIÓN DE CARACTERÍSTICAS MEDIANTE REDES NEURONALES MULTICAPA PARA RECONOCIMIENTO DE IDIOMA

Javier Marcos Quirós
Tutor: Alicia Lozano Díez
Ponente: Joaquín González Rodríguez

Mayo 2017

Extracción de características mediante redes neuronales multicapa para reconocimiento de idioma

AUTOR: Javier Marcos Quirós
TUTOR: Alicia Lozano Díez
PONENTE: Joaquín González Rodríguez

Audio, Data Intelligence and Speech (Audias - ATVS)
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2017

Resumen

Este Trabajo Fin de Grado tiene como objetivo estudiar la utilización de características *bottleneck* extraídas de una red neuronal profunda entrenada para reconocimiento del habla, y reemplazar con ellas a las tradicionales características acústicas MFCC como parámetros de entrada a un sistema de reconocimiento automático de idioma UBM/I-vector.

Para abordar esto, se implementará en primer lugar un sistema de reconocimiento de idioma siguiendo la aproximación clásica UBM/i-vector, basado en características acústicas MFCC que servirá como sistema de referencia. Después se entrenarán cuatro redes neuronales profundas con el objetivo de extraer características *bottleneck* que capturen información a cuatro niveles diferentes de abstracción y finalmente se implementarán cuatro sistemas de reconocimiento de idioma que usen estas características.

La base de datos que se usará para el entrenamiento de las redes neuronales será Switchboard, y para el entrenamiento del sistema de reconocimiento de idioma UBM/i-vector se hará uso de los audios proporcionados por el National Institute of Standards and Technology (NIST) para el Plan de Evaluación de Reconocimiento de Idioma de 2015 (LRE15). Las principales herramientas con las que se trabajarán son: Kaldi para la implementación de los sistemas de reconocimiento de idioma, Theano para el entrenamiento de las redes neuronales y extracción de características *bottleneck*, y también se hará uso de Matlab para generar los modelos, conseguir las puntuaciones (*scores*) y realizar la evaluación de los sistemas.

Para evaluar la mejora que pueda suponer el uso de estas características *bottleneck* en reconocimiento de idioma, se compararán los resultados obtenidos en los cuatro sistemas que usen estas características con los resultados obtenidos con el sistema de referencia basado en MFCC. El rendimiento de los sistemas se medirá según el porcentaje de acierto en la predicción de idioma de los segmentos de evaluación en distintas pruebas previstas. A su vez, se hará una comparativa entre estos sistemas que emplean distintas características de entrada siguiendo la métrica del *Equal Error Rate* (EER) obtenido en cada uno de ellos.

Palabras clave

Reconocimiento automático de idioma, redes neuronales profundas, cuello de botella, UBM, i-vector, Kaldi, Theano.

Abstract

This Bachelor Thesis has the objective of studying the use of bottleneck features extracted from a deep neural network trained for speech recognition in order to replace the traditional MFCC acoustic features as input parameters for an automatic UBM/i-vector language recognition system.

To do so, in the first place a language recognition system with a classical UBM/i-vector approach –based on acoustic features MFCC which works as reference system– will be implemented. After that, four deep neural networks will be implemented with the purpose of extracting bottleneck features that collect information in four different abstraction levels. Finally, four language recognition systems that us this features will be implemented.

The database that will be used for the neural networks training will be Switchboard, and for the UBM/i-vector language recognition system audio recordings provided by the National Institute of Standards and Technology (NIST) for the 2015 NIST Language Recognition Evaluation Plan (LRE15) will be used. The main tools we will be working with are: Kaldi for the language recognition system implementation, Theano for the neural networks training and bottleneck features extraction. Matlab will also be used to generate models, get the scores and perform the systems evaluations.

To evaluate the improvement, these bottleneck features can mean in language recognition, the obtained results will be compared in the four systems that use these features with those obtained with the referral system based on MFCC. The systems' performance will be measured according to the success rate in the language prediction of the evaluation segments over various tests. Moreover, these systems with different input features will be compared following the Equal Error Rate (EER) obtained from each of them.

Keywords

Automatic language recognition, deep neural networks, bottleneck, UBM, i-vector, Kaldi, Theano.

Agradecimientos

Quiero agradecer en primer lugar a mi tutora Alicia Lozano por toda su ayuda durante este trabajo. Y por todo lo pasado en estos años, quiero agradecer todo el apoyo que me han dado mi familia y amigos, especialmente a mis padres, imprescindibles en la forja de quien soy. También quiero agradecer a mi hermana sus imprescindibles correcciones lingüísticas en todo tipo de situaciones y a mi hermano por sus peculiares ánimos para afrontar los exámenes de "unos y ceros". En el ámbito académico y él sabe que en mucho más, a mi amigo Dani, que me ha soportado mis bucles *do while* y todo tipo de implementaciones peligrosas en las prácticas de la carrera. Y además de todos estos, quiero agradecer a quien me escucha siempre con atención cómo los romanos se montaban sus películas y se las hacían creer a los demás.

«Audentis fortuna iuvat»

La Eneida, Virgilio

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación

La señal de voz tiene una especial relevancia en las telecomunicaciones ya que durante mucho tiempo el teléfono fue el principal medio de telecomunicación para la mayoría de población en los países desarrollados. Aunque en el día a día cada vez más personas optan por el uso de aplicaciones de mensajería instantánea o correos electrónicos, el uso del teléfono como tal, aun no siendo el protagonista absoluto como lo fue hasta la progresiva implantación de Internet a toda la red de telecomunicaciones, sí que sigue siendo indispensable.

Las nuevas tecnologías de telecomunicación no son un sustituto de la telecomunicación por voz, sino un complemento. Si bien ha cambiado el canal de transmisión y los dispositivos que se usaban hace cincuenta años, una conversación hablada en algunas ocasiones sigue siendo la mejor manera de comunicarse y en algunos casos como por ejemplo el contacto con distintos servicios de emergencia, la única.

La voz es una señal con mucha y muy distinta información que puede ser usada para una gran variedad de fines. En este trabajo en concreto, nos centraremos en la tarea de reconocimiento automático de idioma (*Automatic Language Recognition*, ALR), que consiste en determinar qué idioma está siendo utilizado en un fragmento de audio. Dicha tarea tiene aplicaciones como pueden ser sistemas de conversación multilenguaje o traductores de lenguaje hablado, y tiene una especial relevancia en materias de seguridad e inteligencia, donde se hace necesario poder establecer qué idioma o idiomas están presentes en las grabaciones antes de extraer cualquier otra información [5].

Por otro lado, los últimos avances obtenidos en el procesamiento digital de voz gracias al uso de redes neuronales profundas (*Deep Neural Networks*, DNNs) [6], invitan a seguir buscando distintas tareas donde el uso de DNNs implique una mejora de rendimiento.

En este Trabajo de Fin de Grado se hará uso de redes neuronales profundas con topología de cuello de botella (*Bottleneck*, BN), es decir, con una de sus capas ocultas significativamente menor al resto. Esto provoca que la información en el *bottleneck* quede en ese punto condensada, haciendo que pueda ser utilizada como representación de la señal para otra tarea. En nuestro caso, la DNN se utilizará por tanto, como extractor de características que ayuden a discriminar entre diferentes idiomas. Lo que se implementará y evaluará será el uso de esta información como parámetro de entrada a un sistema de reconocimiento de idioma.

1.2. Objetivos

El objetivo principal es evaluar el avance que pueda suponer el uso de características *bottleneck* de una red neuronal profunda multicapa con esta topología como entrada a un sistema para reconocimiento de idioma basado en el modelado de la variabilidad total (*Total Variability*, TV), al que denominaremos UBM/i-vector, sistema en el estado del arte para dicha tarea.

La evaluación se realizará comparando los resultados obtenidos a partir de estas características *bottleneck* en reconocimiento de idioma con un sistema UBM/i-vector, con los resultados de este tipo de sistema pero que use como entrada características acústicas basadas en Coeficientes Cepstrales en las frecuencias de Mel (*Mel Frequency Cepstral Coefficients*, MFCC), siendo este el experimento base desde el que se parte.

Puesto que las DNNs se usan como extractores de características BN en este trabajo, el segundo objetivo es evaluar la información extraída según la capa en la que coloquemos el *bottleneck*. Estas redes estarán entrenadas para una tarea del procesamiento del habla, concretamente para clasificación de trifonemas, y lo que se busca al colocar el *bottleneck* en las distintas posiciones es poder observar la relación entre la posición, que implica unas características más o menos dirigidas hacia la discriminación para la tarea de clasificación de trifonemas con una tarea no relacionada, como es el reconocimiento de idioma. Para ello se entrenarán cuatro DNNs con cuatro capas ocultas cuya única diferencia sea la posición del *bottleneck*, cada experimento en una capa distinta.

Adicionalmente y partiendo de los mismos parámetros de salida obtenidos, se buscarán modelados alternativos de estos parámetros que intenten mejorar esta tarea de reconocimiento de idioma.

1.3. Organización de la memoria

La memoria consta de las siguientes secciones:

1-Introducción: Motivación para la realización de este trabajo y objetivos.

2-Estado del arte: Presentación de los fundamentos del reconocimiento de idioma, breve explicación de distintos métodos empleados para esta tarea, e introducción a las tecnologías existentes necesarias para la lectura de la memoria.

3-Entorno experimental: Descripción tanto de las herramientas como de las bases de datos usadas.

4-Desarrollo: Planteamiento del trabajo realizado, desglosado en etapas, fases y funcionalidades. Se explicarán los experimentos realizados así como parámetros de configuración interesantes y conceptos que sean necesarios para su entendimiento.

5-Experimentos y resultados: Exposición de los resultados obtenidos en los distintos experimentos y comparaciones con el experimento base usando distintas medidas de evaluación.

6-Conclusiones y trabajo futuro: Finalmente, se llegará a unas conclusiones en función a los datos obtenidos y propuestas para futuros trabajos acordes a estas.

2. ESTADO DEL ARTE

2.1 Reconocimiento de idioma

El reconocimiento automático de idioma consiste en determinar cuál es el idioma hablado en un segmento de audio. Una persona es capaz de reconocer un idioma en una locución dada bien porque tiene conocimientos de ese idioma, o bien porque reconoce cómo suena, esto quiere decir que es capaz de reconocer algunas de sus características acústicas.

Al igual que el ser humano, el reconocimiento automático de idioma puede por tanto hacerse mediante sistemas que usen características fonéticas o lingüísticas, que englobaremos dentro de los métodos basados en unidades lógicas o *tokens* (*token-based methods*). O también se podría hacer mediante sistemas basados en características del espectro (*spectral-based methods*) representando los segmentos de voz como una concatenación de vectores de características, los cuales mediante un modelado probabilístico y técnicas de clasificación de estos vectores permitirán identificar el idioma [7]. En los dos próximos apartados veremos con algo más de detalle estas dos aproximaciones.

Una ventaja esencial que tiene el reconocimiento automático de idioma es, que a diferencia del ser humano, puede ser expuesto a virtualmente un ilimitado volumen de voz y texto, teniendo el potencial de superar con creces el rendimiento humano en este campo. Para ello es indispensable disponer de bases de datos con un gran número de segmentos de habla y con un rápido y eficiente acceso a estos.

2.1.1 Métodos basados en tokens

La idea en la que se basan estos métodos es en dividir la señal de entrada de habla en unidades lógicas o *token*s, estas unidades pueden estar basadas en segmentación fonética o en segmentación dirigida por los datos. Esta segmentación genera un flujo de *token*s, del cual se extraen las características que serán usadas para clasificar la locución de entrada entre los distintos modelos de idiomas.

Para poder llevar a cabo esta estructura básica y poder dividir correctamente la locución de entrada en distintos *token*s, es imprescindible un etiquetado del habla, ya sea fonéticamente u ortográficamente lo cual limita las bases de datos que pueden ser usadas para estos métodos. Por otro lado, este tipo de aproximaciones engloban un gran número de métodos distintos, entre ellos vamos a destacar a continuación los métodos de reconocimiento de fonemas seguidos de modelado de idioma (*Phone Recognition followed by Language Modelling*, PRLM), que han demostrado un muy buen rendimiento para la tarea de reconocimiento automático de idioma [8][9].

a) *Métodos de reconocimiento de fonemas seguidos de modelado de idioma (Phone Recognition followed by Language Modelling, PRLM)*

Este método reconoce a partir de un audio de entrada unidades de fonemas, tanto de un único idioma como de varios (fonemas multilingües), dividiendo la entrada acústica en unidades lógicas formando una secuencia de etiquetas de estos fonemas [10]. Esta secuencia resultante se introduce en un banco de modelos de distintos idiomas en paralelo, uno por cada idioma a identificar y finalmente, basándose en la puntuación para cada modelo obtenido, un clasificador determina el idioma del audio de entrada.

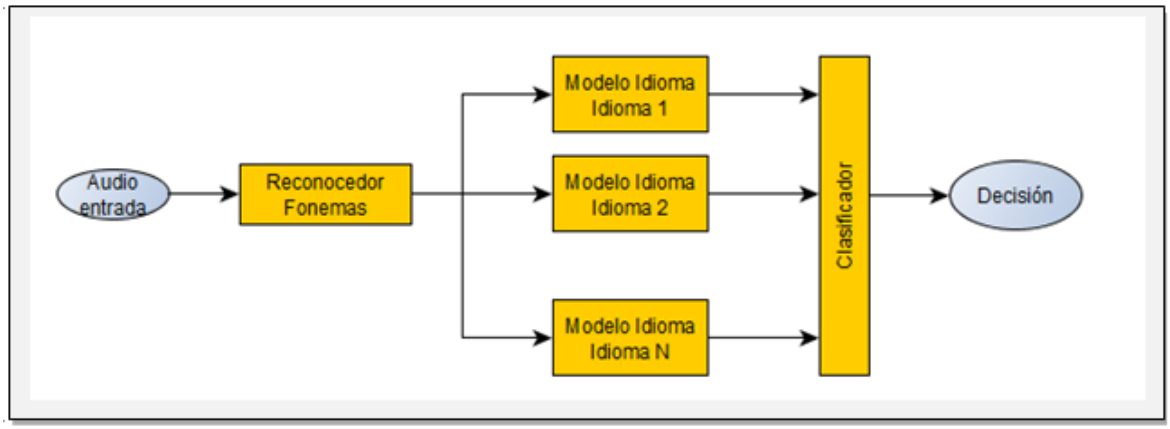

FIGURA 2.1: Phone Recognition followed by Language Modelling, PRLM.

b) *Métodos de reconocimiento de fonemas seguidos de modelado de idioma en paralelo (Parallel Phone Recognition followed by Language Modelling, P-PRLM)*

Este método es una variación del PRLM descrito anteriormente. Un aspecto importante del método PRLM, es que las operaciones de división en unidades lógicas (*tokenization*) y de modelado de idiomas son independientes. No es necesario para el reconocedor de fonemas ser entrenado en ninguno de los idiomas a reconocer para poder realizar la segmentación en unidades lógicas, siempre que los flujos de *token*s resultantes para los estos idiomas sean suficientemente discriminatorios. Por lo que se propuso ejecutar los reconocedores de fonemas para distintos idiomas en paralelo para poder encontrar una mayor diversidad de fonemas [9].

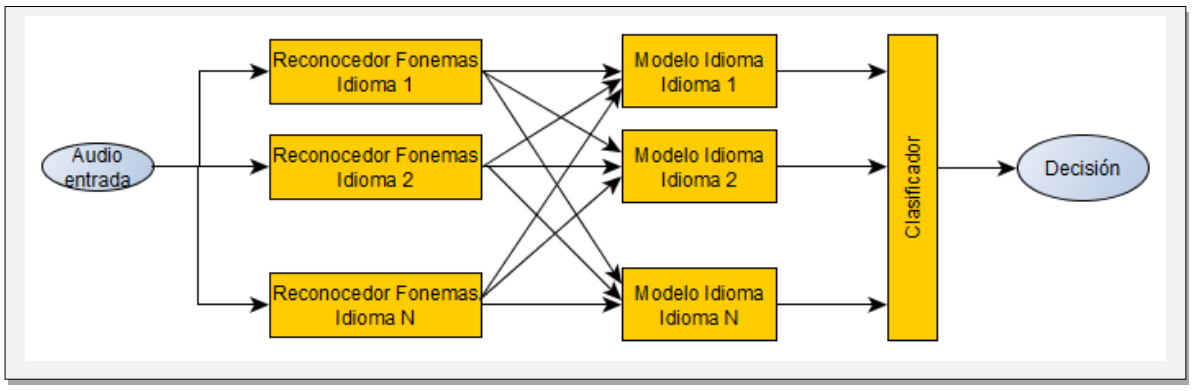

FIGURA 2.2: Parallel Phone Recognition followed by Language Modelling, P-PRLM.

2.1.2 Métodos basados en características del espectro

No siempre se dispone de locuciones con una duración suficiente como para tengan frases o palabras completas, para estos casos es necesario desarrollar unos métodos capaces de distinguir un idioma de otro con muy pocos segundos de duración, incluso con segmentos entre 1-3 segundos se pueden conseguir buenos resultados si se usan características acústicas para generar los modelos de idiomas.

Los métodos basados en características del espectro consiguen modelar los idiomas a partir de las medidas y características espectrales extraídas de ventanas fijas de corta duración, generalmente en torno a los 20-25 milisegundos. A continuación veremos algunas de las técnicas con mejores resultados, que podríamos dividir entre técnicas generativas que intentan representar la distribución de los datos como pueden ser los Modelos de Mezclas Gaussianas (*Gaussian Mixed Models*, GMM), y técnicas discriminativas como Máquinas de Vectores Soporte (*Suport Vector Machine*, SVM) cuyo objetivo es obtener frontera de separación entre las características de estos. Adicionalmente veremos técnicas que intentan compensar la variabilidad de canal o inter-sesión, y la variabilidad de locutor, que en nuestro caso y aplicado al reconocimiento de idioma podríamos llamar inter-idioma o inter-clase.

Estos métodos se basan en características espectrales, típicamente MFCC-SDC (*Shifted Delta Cepstral Features*, SDC). Estos coeficientes cepstrales derivan del análisis del banco de filtros Mel usados también en reconocimiento de locutor, por lo que antes hablar de distintas técnicas y métodos, describiremos brevemente cómo se obtienen estas características espectrales.

- **Shifted Delta Cepstral Features:**

Estas características acústicas basadas en la percepción auditiva humana son ampliamente utilizadas en procesos en los que intervenga el reconocimiento automático de audio, ya sea reconocimiento de habla, de locutor, o como en nuestro caso, de idioma.

Para la obtención de los MFCC [11], en primer lugar se divide el audio en segmentos de corta duración, típicamente en tramas de 20-25 milisegundos de duración solapadas cada 10 milisegundos. Tras un enventanado, típicamente Hamming o Hanning, se obtiene el módulo del espectro de estas tramas aplicando la Transformada Directa de Fourier (*Direct Fourier Transform*, DFT) al que se le aplicará un banco de filtros de escala perceptual Mel como el mostrado en la figura 2.3. Se calculará, del logaritmo de la señal resultante la energía total en cada una de las bandas críticas y finalmente, se aplica una Transformada Directa del Coseno (*Direct Cosine Transform*, DCT), siendo los primeros coeficientes los coeficientes MFCC usados.

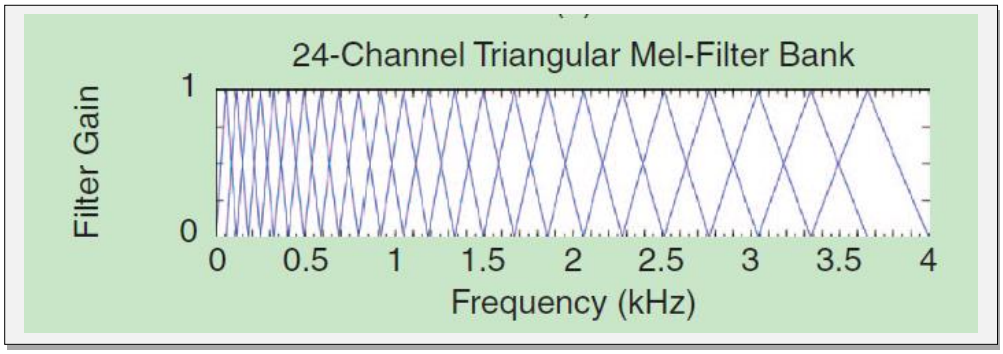

FIGURA 2.3: Banco de filtros triangulares Mel para 24 canales, figura extraída de [1].

El banco de filtros Mel está diseñado para que sea más sensible a variaciones de frecuencia en las partes bajas del espectro gracias a una serie de filtros paso banda triangulares, simulando el sistema auditivo humano [12]. Adicionalmente, parámetros de velocidad (y también de aceleración para reconocimiento de locutor) son calculados a partir de múltiples tramas, conocidos como delta (y delta-delta respectivamente). Las características SDC se forman combinando estos parámetros calculados entre distintos segmentos y son añadidas a los coeficientes MFCC, complementándolos con sus propiedades dinámicas. Esto, junto con las propiedades derivadas de la aplicación de la DCT, que comprime la energía de la señal en unos pocos coeficientes altamente incorrelados, hace de estos coeficientes unas características acústicas de entrada óptimas para sistemas de procesamiento de voz.

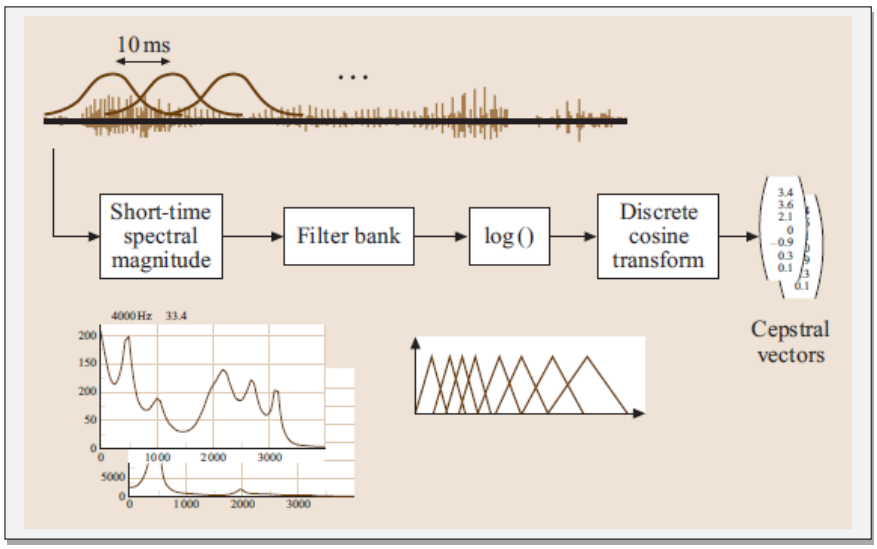

FIGURA 2.4: Extracción de características cepstrales del banco de filtros Mel, figura extraída de [2].

Una vez visto cómo se consiguen las características acústicas que se suelen usar en tipo de métodos basados en características del espectro, veremos en qué consisten empezando por las dos técnicas generativas con más éxito y usadas en el desarrollo de este Trabajo de Fin de Grado: los Modelos de Mezclas Gaussianas (GMM) y una adaptación de esta técnica: el Modelo Universal (*Universal Background Model*, UBM). También mencionaremos la principal técnica discriminativa para métodos basados en características espectrales: las Máquinas de Vectores Soporte (SVM).

TÉCNICAS GENERATIVAS:

a) Gausian Mixture Models (GMM)

Este método modela la distribución de los datos, como pueden ser características cepstrales en reconocimiento de idioma, generando un GMM para cada idioma. Cada GMM tiene una función de distribución que la caracteriza, generada por la suma ponderada de las probabilidades de las distribuciones Gausianas que la forman, siendo esta función de densidad de probabilidad real y multidimensional. Ante un dato nuevo, y tras haber sido entrenado, devolvería la probabilidad de que cada GMM haya generado ese dato, generando un conjunto de puntuaciones que servirán para tareas posteriores de decisión y clasificación.

b) Universal Background Model (UBM)

Conocido también como GMM-UBM por su vinculación con la técnica GMM, son modelos de mezclas Gaussianas que representan a un modelo GMM "universal" permitiendo usar más datos para una misma tarea. En reconocimiento de idioma, el UBM se construye generalmente entrenando un GMM que use un conglomerado de los datos de entrenamiento de todos los idiomas disponibles, creando este modelo universal base para todos los idiomas del que se partirá para crear los modelos de cada idioma por separado adaptando el modelo mediante Maximum A Posteriori (MAP), generalmente adaptando únicamente las medias. Cada GMM adaptado a un idioma contendrá información de ese idioma concreto y también del resto de idiomas usados para generar el UBM, lo que permite una mayor robustez frente a la falta de datos para un idioma. Esto es, la puntuación final dará una idea de cuánto se parece el segmento de test al modelo de idioma, y a la vez, cuánto se distingue del resto de idiomas.

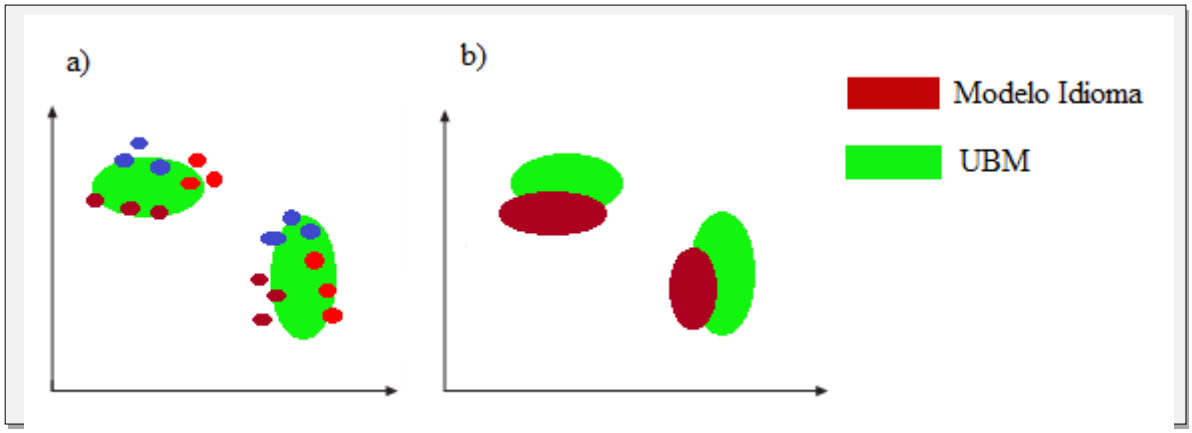

FIGURA 2.5: a) Las contribuciones de todos los idiomas se usan para crear el UBM.
b) Modelo de idioma específico partiendo del UBM.

TÉCNICAS DISCRIMINATIVAS

a) Support Vector Machines (SVM)

Son un algoritmo de clasificación que separa los datos en espacios de alta dimensión usando el de máximo margen, intentando crear una frontera de decisión que maximice el margen o distancia de la frontera a los datos de cada clase, asignando cada patrón a una clase distinta. La configuración general usada para sistemas de reconocimiento de idioma es la de "uno contra todos", mostrada en la figura 2.6.

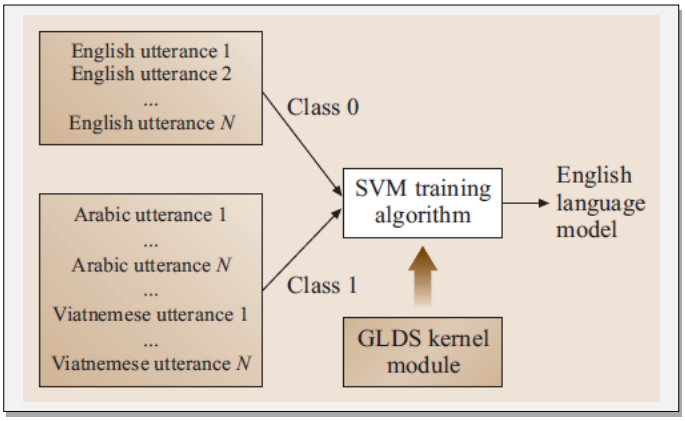

FIGURA 2.6: Entrenamiento de una SVM para reconocimiento de idioma, figura extraída de [3].

2.1.3 Técnicas de compensación de variabilidad y modelo de variabilidad total

Al margen de la división entre técnicas generativas o discriminativas en métodos basados en características espectrales, existen algunas técnicas aplicables a los vectores de características conseguidos con estos métodos, útiles no sólo para el reconocimiento de idioma ya que también pueden ser usados, por ejemplo, para abordar el problema de la variabilidad inter-canal (o más preciso, inter-sesión) en sistemas de reconocimiento de locutor. A continuación veremos técnicas relacionadas entre sí muy importantes en este campo y usadas por tanto en este trabajo.

a) Análisis Factorial (Factor Analysis ,FA)

Al grabar ficheros de audio se almacena información que puede no ser relevante para la tarea buscada o incluso ser perjudicial para un correcto funcionamiento del sistema debido a un sobreentrenamiento enfocado en esa información. Un ejemplo de esto sería un sistema de reconocimiento de locutor en el que un único "locutor A" usase un micrófono específico para las locuciones destinadas al entrenamiento del sistema, y el resto de locutores usasen otro micrófono distinto. Durante este entrenamiento el micrófono diferente proporcionaría características acústicas suficientemente distintivas para que por medio de estas el sistema aprenda reconocer al "locutor A", pudiendo llegar a obviar las características acústicas que si sean útiles para su identificación cuando en las locuciones de evaluación todos los locutores usen el mismo micrófono.

Los métodos generativos comentados anteriormente (GMM, UBM), generan para cada ventana de 20-25 milisegundos un vector de características (supervector) de muy alta dimensionalidad determinado por el número de Gaussianas, que suele ser elevado como el UBM usado en este trabajo que emplea 1024 y el número de características extraídas de la ventana, por ejemplo los 20 primeros MFCC. Este supervector es la concatenación de las medias de las gaussianas del GMM (adaptado del UBM). El análisis factorial trata de capturar la variabilidad de canal en un subespacio de menor dimensión y para poder eliminarla del supervector.

b) Joint Factor Analysis (JFA)

De manera similar a FA, el *Joint Factor Analysis* (JFA) como el usado en reconocimiento de locutor [13], y extrapolable a reconocimiento de idioma, añade el modelado de un subespacio adicional que contenga la información dependiente del idioma. De esta manera, al poder hacer una distinción entre información dependiente del idioma e independiente a este (además del subespacio con la variabilidad de canal), se puede eliminar todavía más información del supervector que no sea relevante para la tarea.

c) Total Variability (TV)

Surgió como solución a los problemas del alto coste computacional de FA y JFA, debido a la necesidad de una gran cantidad de datos de entrenamiento para estimar de forma adecuada los subespacios de variabilidad y a los problemas derivados de usar técnicas de clasificación directamente en supervectores de alta dimensión generados por estas técnicas.

La solución fue crear un único subespacio de baja dimensión llamado espacio de variabilidad total (*Total Variability Space*, T) que contiene al mismo tiempo y aplicado a la tarea de reconocimiento de idioma, la variabilidad del espacio de la información independiente del idioma, y la variabilidad del espacio del canal, pudiendo estimar dicho subespacio de manera apropiada con menos datos [14].

De esta forma, el supervector para un segmento de habla puede descomponerse en la suma del modelo universal UBM más la combinación de este espacio de variabilidad total con un vector de características. Este vector, de menor dimensión que el supervector de las técnicas FA y JFA se le conoce como i-vector.

2.2 Redes neuronales profundas

2.2.1 Introducción: Redes Neuronales y funcionamiento de la neurona

Las redes neuronales son sistemas de aprendizaje automático inspirados en el sistema nervioso biológico, el comportamiento de las neuronas y las conexiones entre estas.

La unidad mínima de computación es la neurona, que la forman una función matemática y sus conexiones de entrada y salida representadas con sus correspondientes pesos. Esta función matemática suele ser de carácter no lineal, como por ejemplo la función *tanh(x)* mostrada en la figura 2.7.a, y su objetivo es a partir de unas entradas y unos pesos

asociados a estas, elegir las salidas correspondientes, también con unos pesos asociados. De forma que una neurona no tiene una misión concreta, si no que con el entrenamiento establecen y refuerzan ciertas conexiones para que una vez entrenada la red, se consiga hacer transformaciones complejas (que incluyen transformaciones no lineales concatenadas) de los datos de entrada, para predecir su salida.

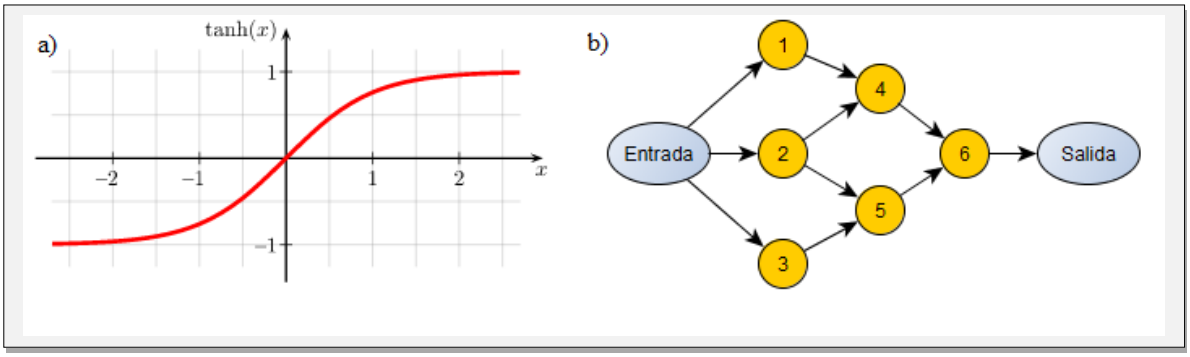

FIGURA 2.7: *a) Función tangente hiperbólica, muy usada en redes neuronales.*
b) Ejemplo de red neuronal sencilla.

Existen numerosos tipos de redes neuronales y con distintas topologías, a continuación veremos los usados en este trabajo: las redes neuronales profundas y la topología de cuello de botella.

2.2.2 Deep Learning, redes neuronales profundas

Las restricciones en el hardware y la falta de grandes cantidades de datos necesarias para entrenar este tipo de redes suponían una barrera infranqueable hasta hace relativamente poco [15]. Las redes neuronales profundas (DNN) son un tipo de configuración en la que las neuronas son colocadas formando varias capas ocultas, y según vayan avanzando cada una de estas capas será capaz de llegar a un nivel mayor de abstracción. Esto quiere decir que en una configuración de este tipo y poniendo como ejemplo una red neuronal profunda con tres capas ocultas cuya tarea sea reconocer rostros de personas en una fotografía, podría describirse su comportamiento (de forma muy general e hipotética) de la siguiente manera:

- La primera capa de neuronas, a partir de una entrada de muy bajo nivel como son los píxeles de una fotografía, sería capaz de extraer aristas o bordes.

- La segunda capa permitiría a partir de la información obtenida en la primera, identificar objetos aislados como ojos, narices y labios.

- La tercera capa teniendo ya la información de estos objetos aislados sería capaz de ubicar en la fotografía dónde están los rostros de personas.

Una red de estas características se entrena de forma iterativa intentando minimizar la función de coste, que evalúa el rendimiento de la red neuronal determinando la diferencia que hay entre la salida predicha y la salida correcta. En el aprendizaje de la red, los pesos asociados a las entradas y salidas de las neuronas de las distintas capas van variando con

cada iteración mediante la técnica del Descenso por Gradiente, que se explicará con más detalle en la sección *4.2.1.Entrenamiento*.

2.2.3 Cuellos de botella

Son un tipo de topología aplicable en las redes neuronales profundas en la que una de las capas ocultas, llamada cuello de botella o *bottleneck* (BN), tiene una dimensión mucho más pequeña que el resto de capas y suele usar una activación lineal, como sucede en las DNNs utilizadas en este Trabajo Fin de Grado. Lo que se busca con esto es que al comprimir la información necesaria para el mapeo de la capa anterior a la capa posterior de este BN, se aumenta la robustez frente al ruido y al sobreajuste (*overfitting*) [16]. Generalmente para lo que se está usando esta tipología es para que las características BN sirvan como entrada a un nuevo sistema, ya sea en una cascada de dos DNNs iguales donde la entrada a la segunda sea precisamente las características BN de la primera red (figura 2.8), o bien que se usen como entrada a otro sistema independiente, como sucederá en este trabajo.

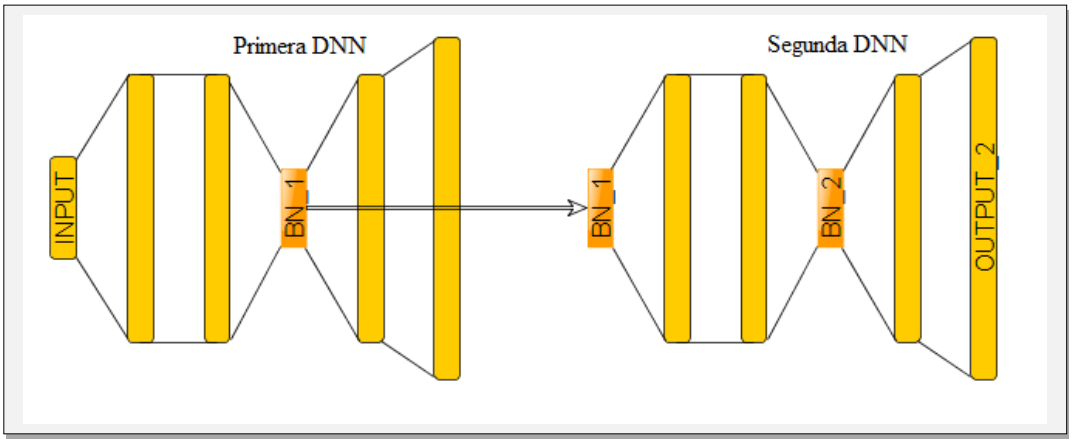

FIGURA 2.8: Diagrama de bloques de una cascada de dos redes neuronales profundas donde las características BN de la primera sirven como entrada a la segunda.

3. ENTORNO EXPERIMENTAL

En esta sección de la memoria se mencionará las herramientas principales y las bases de datos usadas en este Trabajo de Fin de Grado.

3.1 Herramientas

3.1.1 Kaldi

Kaldi [17] es una herramienta de código abierto ampliamente usada en investigación en el campo del procesamiento de voz, escrita en C++ bajo la licencia Apache License v2.0 cuyo objetivo según sus desarrolladores es tener un código flexible, actualizado, y estructurado de manera limpia permitiendo que sea fácil de entender, de modificar y de extender.

Los requisitos buscados para su desarrollo fueron: una infraestructura de transductores de estados finitos (*Finite-State Transducers*, FST), un amplio soporte de álgebra lineal y una licencia no restrictiva, como la licencia Apache v2.0 que acabó usando. Con esto se llegó al desarrollo de Kaldi, que además de las características ya nombradas también tiene otras interesantes de cara a nuestro trabajo: un diseño extensible gracias a que los algoritmos proporcionados intentan ser lo más genéricos posible, y cuentan con ejemplos de sistemas completos (*recipes*) para el uso en sistemas de reconocimiento del habla capaces de trabajar a partir de una gran variedad de bases de datos como las proporcionadas por el Consorcio de Datos Lingüísticos (Linguistic Data Consortium, LDC).

Kaldi se usará en este trabajo para la extracción de características MFCC, y para el entrenamiento de los sistemas de reconocimiento de idioma que usen tanto características MFCC como características BN.

3.1.2 Theano

Theano [18] es una librería de Python que permite definir, optimizar y evaluar expresiones matemáticas con matrices multidimensionales eficientemente, usada ampliamente desde su introducción como compilador matemático tanto en CPU (*Central Processing Unit*) como en GPU (*Graphics Processor Unit*) especialmente por la comunidad de Aprendizaje Automático (*Machine Learning*, ML).

Theano puede automáticamente calcular derivadas simbólicas de expresiones complejas ignorando variables que no sean necesarias para el cálculo del resultado final, reutilizando resultados parciales para evitar cálculos redundantes, aplicando simplificaciones matemáticas, ordenando operaciones de forma que minimice el uso de uso de memoria y aplicando una optimización numérica que regule y minimice el error debido a estas aproximaciones de hardware. Esto lo consigue gracias a que las expresiones matemáticas definidas por el usuario quedan almacenadas como un grafo simbólico de variables y operaciones que será modificado y optimizado durante la compilación.

Aunque Python permite una rápida y fácil interacción con los datos, en muchos casos su intérprete usa un motor insuficiente para ejecutar expresiones matemáticas complejas,

tanto en uso de memoria como en velocidad de cómputo, Theano soluciona estas limitaciones aprovechando lo compacto y maleable que es el lenguaje Python mediante el uso de librerías adicionales.

La Interfaz de Programación de Aplicaciones (*Application Programming Interface*, API) de Theano trata de emular a la librería NumPy, lo que permite al usuario hacer uso de los conocimientos adquiridos para Python mediante una sintáis familiar y una serie de instrucciones avanzadas como el cálculo automático del gradiente, útil para el cálculo del Descenso por Gradiente en el entrenamiento de las redes neuronales profundas usadas en nuestro trabajo. El código que genera es altamente eficiente y permite su uso tanto en CPU como en GPU de manera transparente para el usuario con tan sólo cambiar el valor de una variable de entorno o *flag*.

Theano se usará en este trabajo para el entrenamiento y evaluación de las redes neuronales profundas, y para la extracción de las características *bottleneck*.

3.1.3 Matlab

Matlab es una herramienta de software matemático con un lenguaje de programación propio desarrollado por MathWorks. Ofrece un entorno de desarrollo integrado y es compatible Linux que es la plataforma donde se ha desarrollado este TFG. Matlab permite una manipulación sencilla y eficiente de matrices y por eso lo usaremos en la creación de modelos de idiomas y de obtención de resultados.

Matlab se usará en este trabajo por tanto en la evaluación de los sistemas de reconocimiento de idioma.

3.2 Bases de Datos

3.2.1 The 2015 NIST Language Recognition Evaluation Plan (LRE15)

El Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST) ha llevado a cabo una serie de evaluaciones de la tecnología de reconocimiento de idioma cuyo objetivo es explorar nuevas ideas, y comparar las tecnologías desarrolladas para los problemas que propone, e incentivar el desarrollo de sistemas competitivos [19]. Para la evaluación de reconocimiento de idioma de 2015, se pedía a los participantes que implementaran un sistema que permitiese identificar el idioma presente en un segmento de audio, haciendo hincapié a diferencia de anteriores evaluaciones en idiomas similares que sean fáciles de confundir entre sí. Los veinte idiomas de esta evaluación están agrupados en los siguientes seis clústers, como muestra la tabla 3.1.

Clúster	Idiomas Objetivo
Árabe / Arabic	Egyptian, Iraqi, Levantine, Maghrebi, Modern Estándar
Chino / Chinese	Cantonese, Mandarín, Min, Wu
Inglés / English	British, General American, Indian
Francés / French	West African, Haitian Creole
Eslavo / Slavic	Polish, Russian
Íbero / Iberian	Caribbean Spanish, European Spanish, Latin American Spanish, Brazilian Portuguese

TABLA 3.1: Agrupación de idiomas que forman cada clúster.

Para la realización de este trabajo haremos uso de los datos de entrenamiento proporcionados por NIST para la evaluación, que constan de segmentos de audio extraídos de conversación telefónica (*Conversational Telephone Speech*, CTS) y de radiodifusión de banda estrecha (*Broadcast Narrowband, Speech*, BNBS) en formato 16-bit 8 kHz linear pcm, de una duración de 50 segundos para los segmentos de entrenamiento y de 30 segundos para los segmentos de evaluación.

A partir de estos segmentos se extraerán tanto los coeficientes MFCC como las características BN para el entrenamiento y evaluación de los distintos sistemas de reconocimiento de idioma. Los datos proporcionados por el NIST se dividirán en dos conjuntos distintos para la realización de este trabajo: el conjunto de entrenamiento de cuyos segmentos se usarán para el entrenamiento de los distintos sistemas, y el conjunto de evaluación cuyos segmentos se usarán exclusivamente para la evaluación de estos sistemas. La división de los datos sigue una proporción cercana al 80% para el conjunto de entrenamiento, y el 20% para el conjunto de evaluación, siendo más precisos para el sistema que hace uso de características MFCC se usarán 31.653 segmentos de 50 segundos de duración para el entrenamiento y 8.765 segmentos de 30 segundos para la evaluación.

3.2.2 Switchboard-1 Release 2

Para el entrenamiento de las redes neuronales profundas usadas en este trabajo para la tarea de reconocimiento de trifonemas, se hará uso de la base de datos Switchboard-1 [20], versión 2 [21], recogida por Texas Instruments y actualmente gestionada por el Consorcio de Datos Lingüísticos de la Universidad de Pensilvania. Este corpus consiste en 2438 llamadas telefónicas de seis minutos de duración de media, entre dos locutores en inglés americano. En estos audios de habla telefónica espontánea muestreados a 8 kHz participan más de 500 locutores distintos de ambos sexos

4. DESARROLLO

Los experimentos desarrollados en este Trabajo de Fin de Grado quedarán divididos en tres etapas bien diferenciadas:

Primera etapa: Entrenamiento y evaluación de un sistema UBM/i-vector para reconocimiento de idioma basado en características acústicas MFCC.

Este será el experimento de referencia del que se partirá para evaluar las diferencias y mejoras obtenidas al utilizar características BN, siendo por tanto el objetivo principal de esta etapa tener un sistema de referencia basado en técnicas en el estado del arte para reconocimiento de idioma. Esta etapa sirve como toma de contacto con el entorno así que también se busca ir adquiriendo soltura con las distintas herramientas y bases de datos, entendiendo el funcionamiento de los scripts proporcionados y organizando tanto código como ubicación de ficheros de manera que se pueda automatizar el cambio de parámetros de entrada de manera sencilla para futuros experimentos. Nos referiremos a este experimento como experimento base o *exp_mfcc*.

Segunda etapa: Entrenamiento de cuatro redes neuronales profundas con topología de cuello de botella y extracción de características que serán usadas en el sistema UBM/i-vector de la tercera etapa. La diferencia entre cada entrenamiento será la posición relativa del *bottleneck* en cada una de las cuatro capas ocultas de la red con el objetivo de extraer distintas representaciones (a distintos niveles de abstracción) de la señal de voz.

Estos experimentos son independientes de la anterior etapa y requerirán el uso de GPU para su ejecución en un tiempo razonable. Incluiremos aquí también la extracción de características a partir de los cuellos de botella. Nos referiremos a estos experimentos como: *exp_DNN_BN_1*, *exp_DNN_BN_2*, *exp_DNN_BN_3* y *exp_DNN_BN_4*, por la posición relativa del BN, respectivamente.

Tercera etapa: Se realizarán cuatro experimentos similares al llevado a cabo en la primera etapa, pero usando como parámetro de entrada las características *bottleneck* extraídas en la etapa anterior.

Con esta etapa final se pretende comprobar cómo mejora el rendimiento de este sistema con el uso estas características aprendidas automáticamente por la red, y también la dependencia de los resultados finales con la posición del cuello de botella en cada una de las capas ocultas de la red neuronal profunda. Estos experimentos tendrán los identificadores: *exp_lang_1*, *exp_lang_2*, *exp_lang_3* y *exp_lang_4*.

4.1 Primera etapa: UBM/i-vector MFCC

Dividiremos esta etapa en tres fases distintas: preparación de los datos, entrenamiento del sistema de reconocimiento de idioma y extracción de i-vectors, y finalmente la fase de evaluación de resultados. El código está dividido en diferentes scripts con funciones muy diferenciadas, por lo que para explicar esta primera etapa se agruparán estas funcionalidades y se explicarán por separado, dentro de la fase que corresponda.

El código necesario en esta etapa hará uso de Kaldi para las dos primeras fases y de Matlab para la fase final de evaluación de resultados. Los MFCC usados se extrajeron de las locuciones presentes en la base de datos del Plan de Evaluación de Reconocimiento de Idioma de 2015 (LRE15).

4.1.1 Preparación de características de entrada

El objetivo de esta primera fase es obtener las características de entrada al sistema, mediante la extracción de características MFCC de las locuciones, la aplicación de la Detección de Actividad de Voz (*Voice Activity Detection*, VAD) y la segmentación de locuciones, procesos explicados a continuación:

a) Extracción de características MFCC de las locuciones

Para el experimento de esta primera etapa se hará uso de las características MFCC como parámetro de entrada al sistema, así que el primer paso será extraer estas características.

De cada segmento se extraerán tramas de 20 milisegundos con un solapamiento entre ellas de 10 milisegundos, y se les aplicará un enventanado Hamming. Una vez enventanada la señal y tras aplicar la Transformada Directa de Fourier, al módulo se le aplicará un banco de 25 filtros de escala perceptual Mel [22], con unas frecuencias de corte inferior y superior de 300Hz y 3300Hz respectivamente. Se calculará la energía total logarítmica de cada una de las bandas resultantes de la aplicación del banco de filtros y finalmente se aplicará una Transformación Directa del Coseno, quedándonos para parametrizar un segmento de audio los 7 primeros coeficientes incluyendo C0. Adicionalmente este vector se aumentará con las características SDC, representado su cálculo en la figura 4.1, que han demostrado mejorar significativamente sistemas de reconocimiento de idioma.

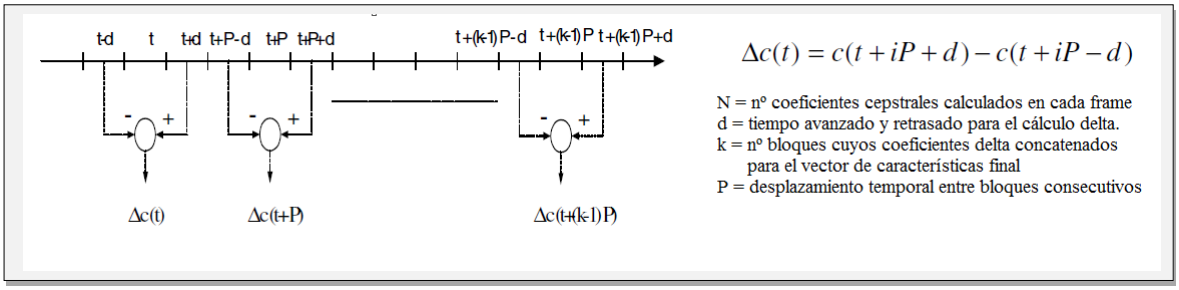

FIGURA 4.1: Cálculo del vector de características SDC en un frame, figura extraída de [4] y definición de parámetros.

b) Aplicación de la Detección de Actividad de Voz

La Detección de Actividad de Voz (VAD) [23] es una técnica usada en aplicaciones de procesamiento de voz que trata de distinguir secciones de voz y de no voz dentro de una señal acústica.

Esto es un paso importante porque permite deshacerse de los segmentos de audio que no aportan información para nuestra tarea de reconocimiento de idioma como los silencios,

que no ayudan a distinguir entre diferentes idiomas. De esta forma almacenaremos y usaremos únicamente los segmentos de locuciones que no sean silencios o pausas, y que contengan por tanto información útil para nuestro sistema.

Para este trabajo usará un VAD basado en una DNN previamente entrenada [24] desarrollado como parte del programa RATS (Robust Automated Transcription of Speech) de DARPA.

c) Segmentación de locuciones

Para este trabajo se han utilizado segmentos de 50 segundos de voz para entrenamiento, de tal manera que contenga suficiente información para la correcta estimación de i-vectors. La evaluación se realiza sobre segmentos de Test de 30 segundos, una de las duraciones de evaluación usadas típicamente en las evaluaciones NIST.

Para poder hacer una correcta comparativa entre los resultados de este experimento *exp_mfcc* con los cuatro experimentos de la tercera etapa, *exp_lang_Nº*, se usarán características extraídas de las mismas locuciones segmentadas para todos los sistemas.

4.1.2 Entrenamiento del sistema de reconocimiento de idioma y extracción de i-vectors (UBM_ivector)

Esta siguiente fase quedará dividida en tres pasos: el entrenamiento del UBM, el entrenamiento del extractor de i-vectors, y la extracción de estos. El número de procesos en paralelo usados variaron en función de la disponibilidad del servidor, afectando al tiempo de ejecución pero no al resultado final.

a) Entrenamiento del UBM

Para el entrenamiento se usarán como características de entrada los vectores finales de dimensión 56 extraídos de los segmentos de entrenamiento, que contienen los MFCC-SDC, en configuración 7-1-3-7. Se usa un UBM completo (sin suposición de covariancias diagonales) entrenado como un GMM, y el GMM-UBM resultante tendrá 1024 gaussianas, conseguidas tras cuatro iteraciones.

b) Entrenamiento del extractor de i-vectors

Tras el entrenamiento del UBM, se extraerán estadísticos a partir las características de entrada (del conjunto de datos de entrenamiento), y con ellos se entrenará el subespacio de variabilidad total, que se puede ver como una matriz de proyección de los supervectores en i-vectors, conteniendo así toda la información en un espacio más reducido, en nuestro caso de dimensión 400.

c) Extracción de i-vectors

Finalmente, se extraerán de los segmentos de entrenamiento y también de los de evaluación i-vectors de dimensión 400 que los caractericen. Son estos i-vectors obtenidos los que se usarán para el modelado de los idiomas a partir de los i-vectors de los segmentos

de entrenamiento, y para la evaluación del sistema a partir de los obtenidos de los segmentos de evaluación, como veremos en el siguiente apartado.

4.1.3 Modelados y evaluación de resultados

Finalmente, esta última fase quedará divida igual que las anteriores en tres funcionalidades: preparación de los datos, creación de modelos, y evaluación de resultados.

a) Preparación de los datos

Para evaluar los resultados será necesario realizar operaciones entre los i-vectors que caracterizan los idiomas y los i-vectors que definan los segmentos de test. Es en esta parte donde se hará uso de Matlab para agrupar los i-vectors en matrices que simplifiquen estas operaciones.

Para relacionar idiomas con segmentos, a cada idioma se le asignará un identificador numérico, estos identificadores son:

Idioma	**ID numérico**	**Idioma**	**ID numérico**
ara-arz	1	eng-usg	11
ara-acm	2	eng-sas	12
ara-apc	3	fre-waf	13
ara-ary	4	fre-hat	14
ara-arb	5	qsl-pol	15
zho-yue	6	qsl-rus	16
zho-cmn	7	spa-car	17
zho-cdo	8	spa-eur	18
zho-wuu	9	spa-lac	19
eng-gbr	10	por-brz	20

TABLA 4.1: Identificadores numéricos para cada idioma.

b) Creación de modelos

Con las matrices de i-vectors y clases del conjunto de entrenamiento, se crean una serie de modelados distintos que a partir de estos datos consigan clasificar un segmento de audio dado entre los veinte idiomas a identificar. Cada modelo distinto tendrá un tamaño de [1x400] (intenta ser un i-vector que caracterice el idioma) y se explicarán con más detalle en el apartado *4.4 Distintos modelados para la evaluación del sistema*.

c) *Scoring*

Finalmente, con las matrices de i-vectors y de clases correspondientes al conjunto de evaluación se comprobará la bonanza de los distintos modelos creados. Para ello se usará el Algoritmo de Puntuación del Coseno (*Cosine Score Algorithm*, CSA) como medida para obtener puntuaciones, y como medidas de rendimiento del sistema se usarán distintas Matrices de Confusión (Confusion Matrix, CM) para la visualización de resultados en la clasificación, además de la Tasa de Error Igual (EER) para un mejor entendimiento de los resultados. Explicaremos a continuación con algo más de detalle estos conceptos:

- **Cosine Score Algorithm**: [25]

Cada idioma a identificar queda representado por un único i-vector modelo de dimensión 400, y cada segmento de test se representa también por un i-vector de 400 dimensiones. Para realizar la identificación del idioma contenido en el segmento de test, éste es comparado con los modelos de los distintos idiomas: ya sea una comparación con todos los idiomas o los restringidos a un clúster. La medida de similitud usada es la basada en el coseno del ángulo que forman el i-vector modelo con el i-vector test.

Un ejemplo con vectores de características con dos únicos datos podríamos verlo en la figura 4.2, tendríamos un espacio vectorial de dos dimensiones en el que si cada modelo de idioma estuviese representado por un vector se podría hacer la clasificación de un segmento desconocido en base al ángulo que forme el vector de las características de este segmento desconocido con los distintos vectores de características de los modelos de idioma.

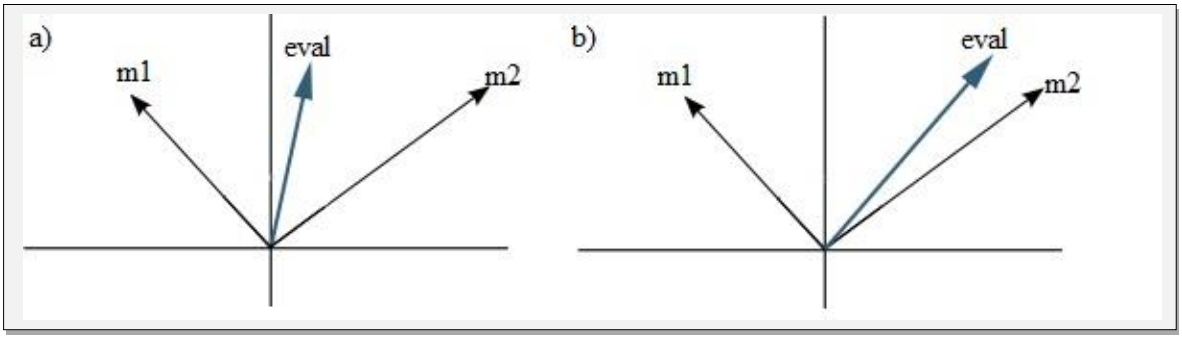

FIGURA 4.2: a) i-vector de evaluación poco concluyente.
b) i-vector de evaluación muy cercano al modelo de idioma 2.

Cuanto más pequeño sea el ángulo que separa ambos vectores más parecidos serán, haciendo posible una clasificación en base a un único valor (score o puntuación). Se hace uso del coseno de este ángulo para crear los parámetros que se usarán para la clasificación final por las siguientes propiedades: Con vectores iguales, el resultado será 1, con resultados opuestos será -1 y con resultados ortogonales será 0. Esto hace que sea muy cómodo de cara a implementarlo en código, cuanto más se acerque la puntuación a 1 más similar será el segmento al modelo.

En nuestro caso, tenemos un espacio vectorial de 400 dimensiones, y podremos conseguir unas puntuaciones de forma eficiente mediante el uso de matrices. La puntuación de similitud entre el segmento y el modelo vendrá dado por la operación:

$$Score = \frac{W_{lang} * W_{test}}{||W_{lang}|| * ||W_{test}||}$$

Donde Wlang es el i-vector modelo de idioma y Wtest es del segmento de test.

Para obtener todas las puntuaciones de todos los segmentos con todos los idiomas (figura 4.3), se multiplicará la matriz de i-vectors de los segmentos de evaluación, de tamaño [nº Segmentos x 400] por la matriz de modelos con la que queramos comparar, de tamaño

[400 x nº Modelos], teniendo como resultado una matriz de tamaño [nº Segmentos x nºModelos] en la que cada fila son las distintas puntuaciones obtenidas para ese segmento, considerando para la decisión de clasificación en idiomas la más alta de ellas.

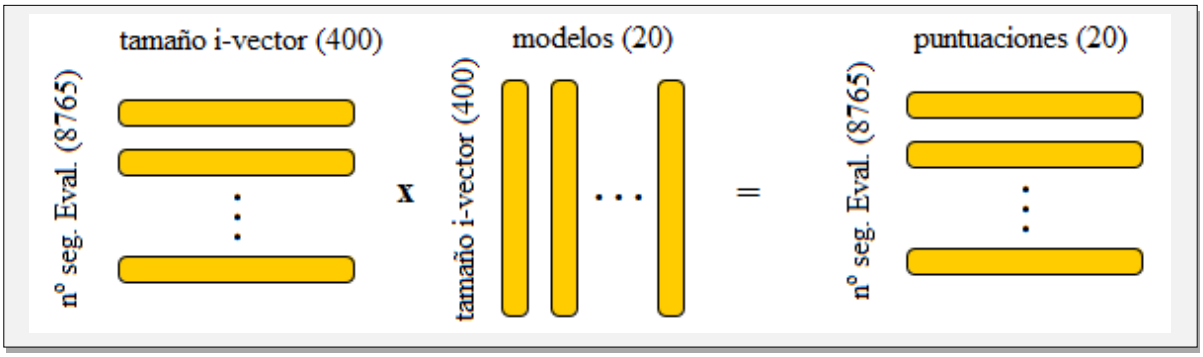

FIGURA 4.3: Obtención de puntuaciones mediante multiplicación de matrices.

Tras obtener las puntuaciones, estas se usarán para medir el rendimiento del sistema en los modelados que se explicarán en la sección *4.4 Distintos modelados para la evaluación del sistema,* así como para la evaluación según el EER. Cada segmento de evaluación tendrá tantas puntuaciones como enfrentamientos se hayan realizado durante su obtención (por ejemplo veinte scores distintos en el caso de enfrentamiento total), y se elegirá el idioma que tenga la mayor puntuación como aquel más probable para ese segmento. Para medir el rendimiento del sistema usando esta clasificación se almacenarán las decisiones en matrices de confusión, explicadas a continuación. Para la evaluación según el EER se usarán directamente estas puntuaciones ya que sirve para medir el error de verificación sin haber tomado ninguna decisión.

- **Confusion Matrix**:

Una vez clasificadas las locuciones, no sólo nos interesa llegar a un porcentaje de acierto, es interesante además ver en los segmentos clasificados erróneamente por qué idioma se confundió, por lo que el uso de Matrices de Confusión, también llamadas tablas de contingencias (*contingency tables*) [26] será de gran ayuda tanto para la visualización como para posteriores cálculos de medidas de error, especialmente en nuestro problema, donde analizaremos por separado clústers de idiomas similares.

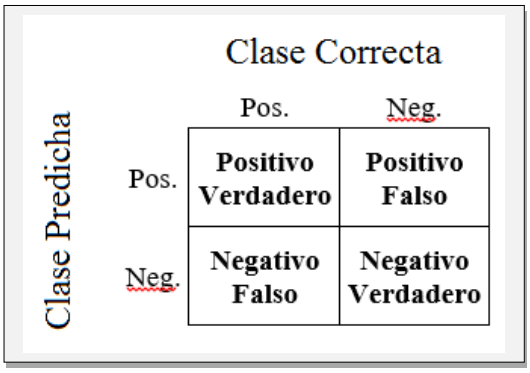

FIGURA 4.4: Matriz de confusión para un sistema binario.

Básicamente una matriz de confusión almacena todas las clasificaciones hechas, tanto las acertadas como las falladas, mostrando cuantos segmentos de un idioma se confundieron con otro, y permitiendo, al estar en forma de matriz realizar operaciones con estos resultados de manera sencilla. En un sistema binario estas clasificaciones serían:

- Positivo Verdadero (*True Positive*, TP): Si la clase es positiva y se clasifica como positiva.
- Positivo Falso (*False Positive*, FP): Si la clase es negativa y se clasifica como positiva.
- Negativo Verdadero (*True Negative*, TN): Si la clase es negativa y se clasifica como negativa.
- Negativo Falso (*False Negative*, FN): Si la clase es positiva y se clasifica como negativa.

En nuestro sistema cada clase será un idioma, así que extrapolando este concepto, una clasificación puede ser:

- Acierto: La predicción de idioma para el segmento coincide con su etiqueta.
- Fallo: El segmento ha sido confundido con otro idioma

De forma que habrá tantas posibles clasificaciones como idiomas enfrentados entre sí. En un enfrentamiento total entre todos los segmentos e idiomas, habrá veinte distintas, mientras que en un enfrentamiento por ejemplo entre segmentos pertenecientes al clúster eslavo pasará a ser un sistema binario con la clase Idioma Ruso y la clase Idioma Polaco.

De forma que aunque puede haber tantas posibles clasificaciones como idiomas enfrentados entre sí, todas las clasificaciones erróneas se condensarán en una sola, teniendo por tanto un sistema binario.

- **Equal Error Rate**:

Las evaluaciones de los distintos experimentos se harán buscando como tarea final del sistema la clasificación de un segmento entre los distintos idiomas a enfrentar. No obstante, para evaluar el rendimiento del sistema sin tener que elegir un umbral de decisión, que en nuestro caso será escoger el idioma que proporcione la puntuación más alta, una medida de evaluación interesante es el EER, que es en el punto en el que las tasas de falsa aceptación y falso rechazo se igualan. Se mostrarán los resultados de los EER de los enfrentamientos de un segmento a los idiomas de su mismo clúster (que es lo que se pedía en la evaluación de sistemas de reconocimiento de idioma NIST LRE 2015 [19]) y la media de los EER de cada clúster, tomando como objetivo (*Target*) el idioma al que corresponde el segmento y no-objetivo (*Non-Target*) el resto de idiomas del clúster. Se obtendrá por tanto, un EER medio de cada problema "*one-vs-all*" (un frente a todos).

4.2 Segunda etapa: DNN-BN

Esta etapa del trabajo quedará dividida en dos pasos: primero se entrenará la red neuronal profunda para la tarea de reconocimiento automático del habla (*Automatic Speech Recognition*, ASR) para, una vez entrenada, utilizarla como extractor de características para los datos de LRE15, que se usarán para el desarrollo de la tercera etapa UBM/i-vector-BN.

4.2.1 Entrenamiento

La red entrenada tendrá la misma estructura que la usada en los experimentos llevados a cabo en [27], donde se aplica a reconocimiento de locutor. Consiste en una red neuronal profunda con topología de cuello de botella, entrenada para reconocimiento de estados de trifonemas. Para este entrenamiento se usará la base de datos Switchboard y unas etiquetas de los estados de trifonemas que vienen de un sistema GMM-HMM (*Hidden Markov Models*, HMM) desarrollado en Kaldi y que han sido aportadas por la tutora.

Lo que se busca con esta topología de cuello de botella es condensar la información en un vector de características de dimensión reducida, una capa *bottleneck* con 80 unidades en nuestro caso, con el objetivo de obtener una información útil para la discriminación de idiomas posterior como ya se comentó en el apartado *2.2.3 Cuellos de Botella* de esta memoria. La red recibirá como entrada un vector de dimensión 120 (se tomarán los 6 primeros coeficientes de la DCT por cada uno de los 20 MFCC) con la información MFCC del frame central y de los 15 frames anteriores y posteriores de contexto tras un preprocesamiento DCT del segmento, y tendrá como salida 3.083 unidades, que mostrarán la probabilidad del segmento de entrada de contener cada uno de los estados de trifonemas. Esta red se entrenará de manera supervisada, obteniendo unos pesos entrenados para esta tarea de clasificación en estados de trifonemas.

Esta red, representada en la figura 4.5, tiene en total cuatro capas ocultas, tres de ellas de 1.500 valores o neuronas con una función de activación de sigmoide, y el cuello de botella con una función de activación lineal, que se irá colocando en distintas posiciones según el experimento, tendrá 80. Sus identificadores serán los siguientes:

- *exp_lang_1*: Cuello de botella en la primera posición de cuatro.
- *exp_lang_2*: Cuello de botella en la segunda posición de cuatro.
- *exp_lang_3*: Cuello de botella en la tercera posición de cuatro.
- *exp_lang_4*: Cuello de botella en la cuarta posición de cuatro.

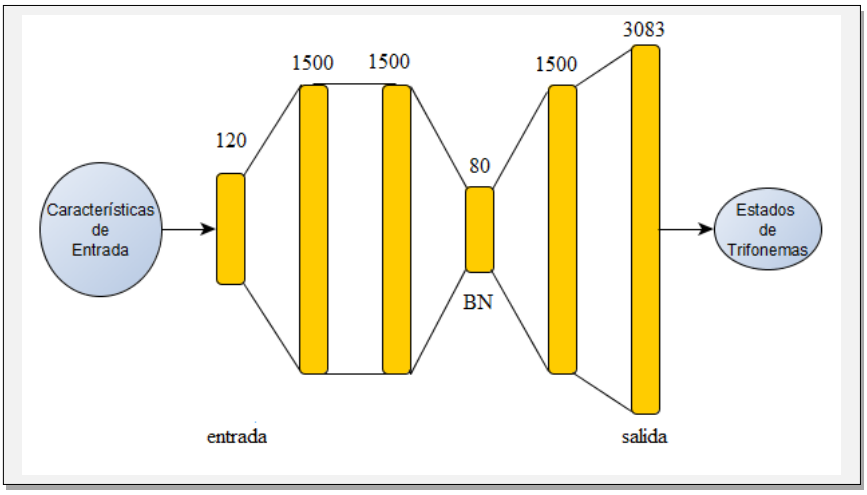

FIGURA 4.5: Ejemplo de la estructura de la DNN usada, con el BN en la tercera posición.

El entrenamiento de las DNNs se llevará a cabo por medio del Descenso por Gradiente:

- **Descenso por gradiente:** [28]

Este método parte de una inicialización, normalmente aleatoria, de los pesos de entrada y una propagación hacia delante por las capas ocultas de la red, generando una salida, que se compara con la salida buscada obteniendo un error de acuerdo a una función de coste que se pretende minimizar (figura 4.6). Tras esto se realiza una propagación hacia atrás del error, partiendo de estas salidas para acabar en la entrada a la red, calculando las derivadas parciales que describen la aportación de cada neurona y conexión al error total. Para minimizar el error de clasificación de la tarea para la que se entrena, se deberá por tanto minimizar estas aportaciones a la señal de error total, repitiendo iterativamente esta minimización de error hasta que los pesos iniciales converjan a los pesos definitivos de la red.

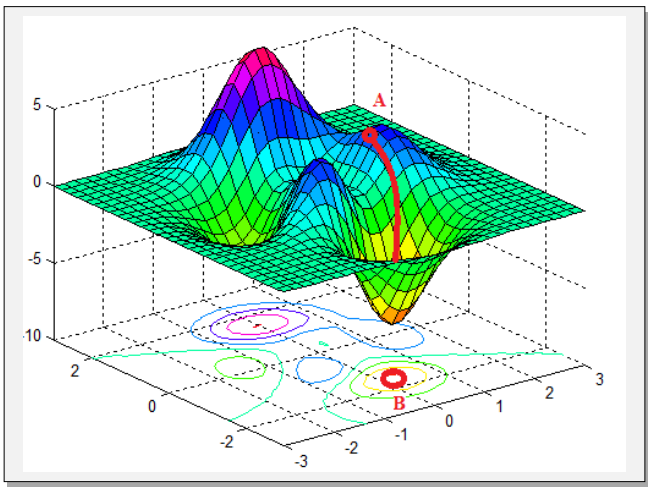

FIGURA 4.6: Representación del Descenso por Gradiente sobre la función de ejemplo Peaks de Matlab, parte desde una inicialización aleatoria en A y acaba en el mínimo B.

4.2.2 Extracción

Una vez entrenada la red, (recordemos, una red por cada experimento) el siguiente paso es extraer de los segmentos de entrenamiento y de evaluación de la base de datos LRE15 los valores de las activaciones de las neuronas de la capa que caractericen estos segmentos. Esto se consigue propagando los 20 primeros MFCC a través de la red entrenada, pero acabando en el cuello de botella y extrayendo esos valores, sin finalizar el recorrido de la red hasta la clasificación final. Siendo los vectores resultantes de este proceso los que sustituirán en los experimentos de la tercera etapa UBM/i-vector-BN a los MFCC del experimento base.

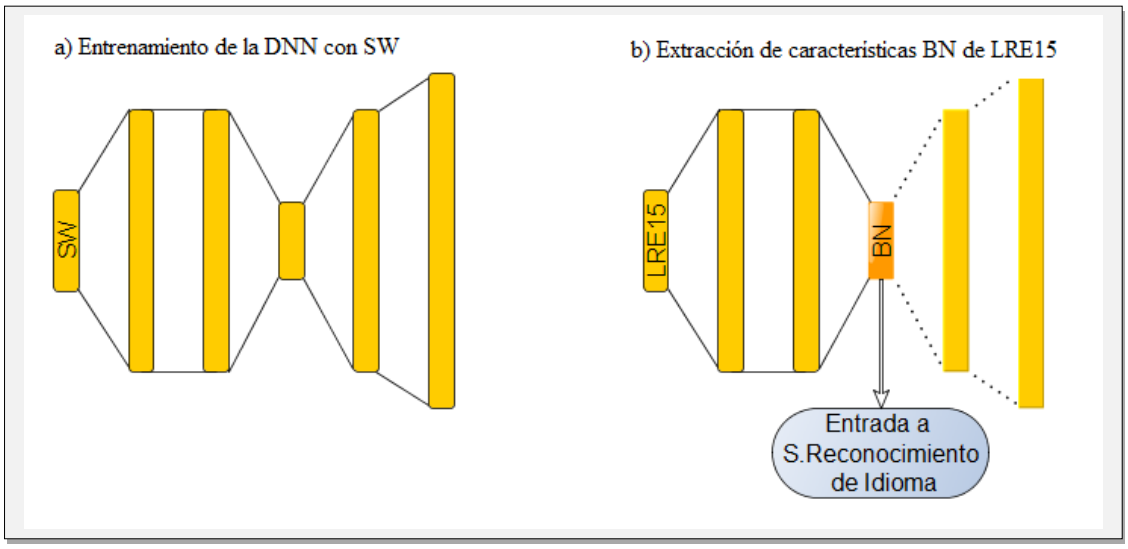

FIGURA 4.7: Tras el entrenamiento de la DNN con los datos de la base de datos Switchboard (a), se extraerán las características BN de los datos LRE15 tras su paso por la red, si necesidad de pasar por todas las capas ocultas (b).

4.3 Tercera etapa: UBM/i-vector BN

En esta tercera y última etapa se repetirá el experimento base realizado, usando las características de *bottleneck* extraídas en la anterior etapa en lugar de las características MFCC de los segmentos de voz como ya se ha explicado en apartados anteriores. Se realizarán cuatro experimentos, uno por cada red entrenada y su identificador corresponderá, igual que en la red, a la posición que ocupe el cuello de botella teniendo por tanto los mismos identificadores: *exp_lang_1*, *exp_lang_2*, *exp_lang_3* y *exp_lang_4.*

Estos experimentos usarán las mismas herramientas que el experimento *exp_mfcc*, y a excepción de la primera fase de este, *4.1.4 Preparación de características de entrada*, usarán como base el mismo código. Como entrada al sistema UBM/i-vector se usará una lista con las características *bottleneck*, y a partir de este punto los pasos posteriores serán iguales a los del *exp_mfcc* explicados en las secciones *4.1.2 Entrenamiento y Extracción de i-vectors* y en *4.1.3 Modelados y evaluación de resultado*, usando las características BN en vez de características acústicas MFCC.

4.4 Distintos modelados para la evaluación del sistema

En este apartado de la memoria se explicarán los distintos modelados de idioma generados para los experimentos, así como los objetivos de las pruebas realizadas para poder interpretar correctamente sus resultados.

Por una parte, se buscaba comprobar la eficacia del sistema enfrentando únicamente segmentos de idiomas pertenecientes a un mismo clúster de idiomas. Esto es, dividir los segmentos del conjunto de evaluación y enfrentar cada segmento perteneciente a un clúster con los modelos de idioma de ese clúster. Por otra parte, también se buscó examinar la capacidad del sistema para identificar correctamente el idioma de un segmento enfrentándolo a los veinte modelos de idioma disponibles.

Para el sistema UBM/i-vector usado en este Trabajo Fin de Grado, un segmento de audio queda caracterizado por un i-vector de 400 valores, por lo que los modelos de idiomas que se generarán a partir de esta información serán otro i-vector de 400 dimensiones. Cada uno de los veinte modelos de idioma fue generado haciendo la media de todos los segmentos de entrenamiento pertenecientes a ese idioma. Estos modelos se agruparon en distintas matrices para la aplicación del Algoritmo del Coseno explicado en la sección *4.1.3 Modelados y evaluación de resultados* de la memoria.

-*Matrices Por Clústers*: Los modelos se agrupan en seis matrices distintas: árabe, china, inglesa, francesa, eslava e íbera, lo que al separar por clústers también los segmentos de evaluación permitirá que cada segmento de audio se enfrente únicamente a los modelos de idioma del clúster al que pertenece.

-*Matriz de Enfrentamiento Total*: Esta matriz contendrá el total de modelos de forma que cada segmento se enfrentará a los veinte modelos distintos y elegirá como idioma predicho el que mayor puntuación obtenga.

En el experimento base se observó que la precisión de los enfrentamientos por clústers era entre un 4% y un 14% superior a los enfrentamientos totales, donde los fallos interclúster bajaban el rendimiento. Asumiendo que esto se repetiría para los siguientes experimentos e intentando generalizar al caso en el que no se conozca de antemano a qué clúster pertenece un segmento de test, se intentó entonces hacer dos aproximaciones que aprovechasen los mejores resultados intraclúster de cara a un enfrentamiento total. Para ello se generarían *Modelos de clúster*, donde se caracterizase no un idioma, si no todos los idiomas presentes en cada clúster. Estas dos aproximaciones son:

Implementación Alternativa A: Cada modelo de clúster se generará haciendo la media de todos los segmentos pertenecientes a ese clúster. Los idiomas con mayor número de segmentos de entrenamiento tendrán por tanto mayor peso en el modelo creado.

$$M^{cluster} = \frac{Lang_1 1 + Lang_1 2 + \cdots + Lang_1 S_1 + \cdots + Lang_L S_L}{S_1 + S_2 \ldots + S_L}$$

Donde $Lang_L S$ indica el i-vector del segmento S del idioma L, y S_L el número de segmentos (i-vectors) disponibles para el idioma L, siendo $M^{cluster}$ el modelo de clúster.

Implementación Alternativa B: Cada modelo de clúster se generará haciendo la media de todos los modelos de idioma pertenecientes a ese clúster. En este caso, los datos de cada idioma se ponderarán de manera diferente para que la aportación de todos los idiomas al modelo de clúster sea la misma.

$$M_L^{lang} = \frac{Lang_L 1 + Lang_L 2 + \cdots + Lang_L S_L}{S_L}$$

$$M^{cluster} = \frac{M_1^{lang} + M_2^{lang} + \cdots + M_L^{lang}}{C}$$

Donde M_L^{lang} es el modelo del idioma L formado por los i-vectors LangLSL siendo SL el número total de i-vectors para ese idioma L, y $M^{cluster}$ el modelo de clúster para un número C de idiomas en el clúster.

De forma que para predecir el idioma de un segmento de evaluación en estas implementaciones alternativas se hará en dos etapas, como muestra el ejemplo de la figura 4.8.

Primero se enfrentará con los seis modelos de clúster, siendo el que obtenga la puntuación más alta el clúster que contenga el idioma más probable. En segundo lugar, se enfrentará el segmento a los modelos de idiomas pertenecientes al clúster ganador del primer enfrentamiento.

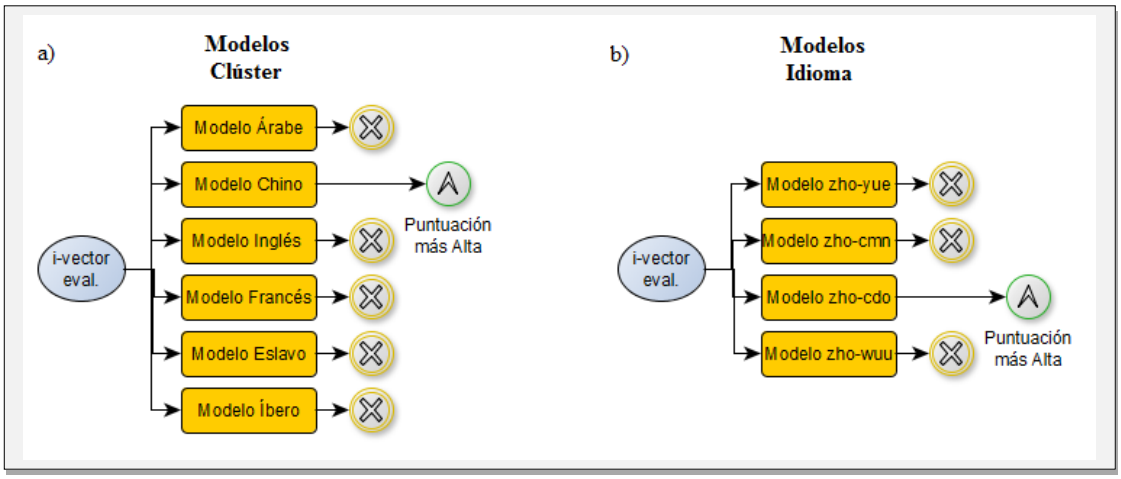

FIGURA 4.8: Enfrentamiento en dos etapas, un primer enfrentamiento con los modelos de clúster (a), para un posterior enfrentamiento con los modelos de idioma perteneciente al clúster con mayor puntuación (b).

Por tanto, para uno de los cinco experimentos UBM/i-vector se realizarán tres evaluaciones distintas de enfrentamientos totales: Enfrentamiento libre veinte contra veinte, usando la Implementación Alternativa A y usando la Implementación Alternativa B. Y también por cada experimento se realizarán seis pruebas de enfrentamientos por clúster: árabe, chino, inglés, francés, eslavo e íbero. Teniendo por tanto 9 enfrentamientos distintos por experimento y un total de 45 para observar con mejor perspectiva cómo afecta el cambio de los parámetros de entrada a este sistema de reconocimiento de idioma al usar, en vez de características acústicas MFCC usar características BN de una red entrenada para otra tarea distinta.

5. EXPERIMENTOS Y RESULTADOS

En este apartado de la memoria se expondrán los resultados más relevantes de los distintos experimentos hechos durante el desarrollo de este trabajo, pudiendo encontrar la totalidad de los resultados en los Anexos, en forma de matrices de confusión.

5.1 Experimentos UBM/i-vector MFCC

El método de evaluación para comparar este experimento base con los experimentos de la tercera etapa, UBM/i-vector BN, será mediante el índice de acierto del sistema. Este método contempla que el idioma predicho para un segmento de evaluación será aquel que obtenga la puntuación más alta tras aplicar el algoritmo del coseno explicado en el apartado *4.1.3 Modelados y evaluación de resultados* de esta memoria.

Los siguientes resultados que muestran la tabla 5.1 corresponden a los índices de acierto de los enfrentamientos expuestos al final del apartado *4.4 Distintos modelados para la evaluación del sistema*.

Enfrentamiento Total (a)	20vs20	I.A - A	I.A - B
	83.2744	81.4375	58.243

Enfrentamiento por clústers (b)	arab	chin	eng	fran	slav	iber	Media Clúster
	90.534	90.5235	93.7636	97.2222	96.206	87.4797	92.6215

TABLA 5.1: Índices de acierto respecto al 100% para los enfrentamientos totales (a) y para los enfrentamientos por clúster (b) en el experimento base.

En el enfrentamiento de cada segmento a los veinte modelos de idiomas distintos se obtiene un índice de acierto del 83.2744%, y en los enfrentamientos por clústers el idioma predicho para un segmento tiene de media un índice de acierto del 92.6215%. Las implementaciones alternativas intentaron hacer uso de esta mejor distinción entre idiomas pertenecientes a un mismo clúster (tabla 5.1.b) creando modelos de clúster que permitiesen orientar la clasificación hacia este enfrentamiento (segundo enfrentamiento de las evaluaciones alternativas).

Estas implementaciones alternativas no consiguieron superar el índice de acierto para los enfrentamientos totales, no obstante, la Implementación Alternativa A se acerca bastante con un 81.4375% de segmentos acertados correctamente, y lo más interesante, para algunos idiomas mejora el número de predicciones correctas. La tabla 5.2 de la siguiente página muestra la diferencia entre la matriz de confusión de la Implementación Alternativa A y la matriz de confusión del enfrentamiento 20vs20, pudiendo comparar así las predicciones hechas por cada implementación (cuyas matrices de confusión se encuentran en los anexos) de manera sencilla.

Ante los resultados obtenidos se buscará comprobar en los siguientes experimentos si esta Implementación Alternativa A sigue mejorando la predicción en algunos idiomas, y de ser así, buscar una relación entre esos idiomas y una posible explicación.

I.A-A - 20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-126	0	0	1	-1	-1	2	1	-1	-3	14	-1	13	62	9	4	33	4	11	-21
ara-acm (2)	0	-36	-1	0	-9	-11	0	-2	-2	-2	1	-1	36	29	1	5	0	-5	-1	-2
ara-apc (3)	0	11	2	6	4	-3	0	-8	1	-2	0	0	4	8	6	2	0	-19	-12	0
ara-ary (4)	0	-1	-4	-29	0	-1	0	-4	-2	-1	0	1	9	16	10	17	2	-2	0	-11
ara-arb (5)	0	0	0	0	-33	0	0	0	0	-1	0	0	3	30	1	0	0	-1	1	0
zho-yue (6)	0	0	0	0	0	-7	0	0	0	0	0	0	2	0	0	1	0	2	2	0
zho-cmn (7)	-1	0	0	0	0	0	19	0	0	-3	2	-2	-7	-1	0	0	0	-1	-1	-5
zho-cdo (8)	0	1	0	0	0	-3	0	-19	-4	0	0	1	17	7	0	1	0	-1	0	0
zho-wuu (9)	0	-1	0	0	0	-1	0	0	-4	0	0	0	4	0	0	0	0	0	2	0
eng-gbr (10)	0	0	0	0	-1	0	0	0	0	-3	0	0	0	3	1	0	0	0	0	0
eng-usg (11)	0	-1	0	0	-3	-2	0	-16	-6	1	59	13	-2	0	-2	-2	-10	-19	1	-11
eng-sas (12)	0	0	-5	0	0	-2	0	-3	-2	0	0	-9	8	9	5	6	0	0	-6	-1
fre-waf (13)	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	-17	0	0	0	-4	-2	0	-6	-14	4	42	5	0	-4	-4	0
qsl-rus (16)	0	-1	0	0	-3	0	0	-1	-1	-3	0	0	3	4	1	6	0	-3	0	-2
spa-car (17)	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	-3	0	0
spa-lac (19)	0	0	0	0	0	-3	0	-1	0	0	0	0	0	0	0	-1	0	1	4	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-8

TABLA 5.2: Corresponde a la resta de la matriz de confusión correspondiente al enfrentamiento 20vs20 a la matriz de confusión correspondiente a la Implementación Alternativa A, quedando como resultados positivos en la diagonal aquellos idiomas en los que la Implementación Alternativa A sea superior, y en negativo los idiomas en los que una comparación total con todos los modelos de mejor resultado. Las casillas con un 0 habrán acertado el mismo número de segmentos ambas implementaciones y los valores fuera de la diagonal indican diferencias entre errores, independientemente de su signo.

5.2 Experimentos DNN-BN

Para las redes entrenadas en la segunda etapa se buscaba comprobar la relación existente entre la posición del *bottleneck* y las características derivadas de esta posición usadas para los experimentos de la tercera etapa. Los resultados de la tabla 5.3 muestran los índices de acierto en la clasificación de trifonemas, tanto del conjunto de entrenamiento como de validación de los cuatro experimentos, según la posición relativa del *bottleneck* en la red.

Accuracy	exp_DNN_BN_1	exp_DNN_BN_2	exp_DNN_BN_3	exp_DNN_BN_4
Train	0.524191	0.516667	0.517095	0.507407
Validation	0.491309	0.494811	0.495207	0.481212

TABLA 5.3: Índices de acierto en la clasificación de trifonemas para las DNNs entrenadas.

Se puede apreciar que los resultados son muy parejos en la precisión para el conjunto de validación, que es como evaluaremos la bonanza de los resultados, siendo ligeramente superior el experimento *exp_DNN_BN_3* con el *bottleneck* en la tercera capa oculta sobre cuatro con un 49.5207% de los segmentos clasificados correctamente.

Aunque la posición del *bottleneck* no afecte de manera determinante en la tarea de clasificación de trifonemas usando esta red específica y los datos de la base de datos Switchboard, veremos en siguientes apartados cómo sí que influye de manera más significativa en las características extraídas y los resultados obtenidos a partir de ellas para la tarea de reconocimiento de idioma.

5.3 Experimentos UBM/i-vector BN

Tras el entrenamiento de las cuatro redes neuronales y la extracción de las características BN en la anterior etapa, se ejecutaron los cuatro experimentos finales. Los índices de acierto de los enfrentamientos totales fueron los siguientes:

Enfrentamiento total	20vs20	I.A - A	I.A - B
exp_lang_1	83.0536	78.8284	54.779
exp_lang_2	87,1303	84,2298	60.9455
exp_lang_3	90.9786	89.3457	66.4725
exp_lang_4	92.3261	89.8139	68.2996

TABLA 5.4: Índices de acierto en los enfrentamientos totales (20vs20) para los cuatro sistemas de reconocimiento de idioma entrenados con características BN.

Al igual que en el experimento base *exp_mfcc*, un enfrentamiento entre el segmento y todos los modelos de idioma sigue dando mejor resultado que el uso de las implementaciones alternativas, aumentando en el mejor de los casos (*exp_lang_4*) el índice de acierto de 83.2744% a 92.3261%. La Implementación Alternativa A también consigue una mejora notoria, pasando de un 81.4375% a un 89.8139% de segmentos acertados correctamente y la Implementación Alternativa B, aunque mejorando un 10% los

resultados obtenidos en *exp_mfcc* no consigue unos resultados tan buenos como las otras opciones.

Respecto a la mejora en el índice de acierto de varios idiomas en la Implementación Alternativa A comentada en el apartado *5.1 Experimentos UBM/i-vector MFCC* y en la tabla que representa la resta de dos matrices de confusión (tabla 5.2), es interesante comentar que hay varios idiomas, los presentes en la tabla 5.5, para los que en todos los experimentos se aumenta el número de segmentos acertados correctamente.

Idioma	exp_lang_1	exp_lang_2	exp_lang_3	exp_lang_4	Media	Seg.Totales	Mejora (%)
zho-cmn	32	21	19	12	21	1305	1.6091954
eng-usg	80	66	59	40	61,25	1709	3.58396723
qsl-pol	36	36	42	48	40,5	474	8.5443038
qsl-rus	29	24	6	2	15,25	264	5.77651515
spa-car	4	3	1	1	2,25	460	0.48913043

TABLA 5.5: Incremento de segmentos correctamente clasificados y su porcentaje respecto al número total de segmentos de cada idioma.

Cómo los resultados fluctúan dependiendo de la posición del *bottleneck*, para medir relevancia de estos se usó la media de los cuatro experimentos y el número de segmentos totales de evaluación correspondientes a estos idiomas para comprobar si esta mejora es realmente significativa o no.

Para este criterio, los idiomas pertenecientes al clúster eslavo, *qsl-pol* y *qsl-rus*, obtienen unas mejoras del 8,5443% y 5,7765% respectivamente. Aunque significativo, hay que tomar este dato con cautela ya que el número total de segmentos de estos idiomas no es muy alto comparado con otros.

Vistos los resultados para los enfrentamientos totales, ahora se comprobarán los resultados en los enfrentamientos por clúster, donde el segmento se enfrenta únicamente a los modelos pertenecientes a su mismo clúster.

Enfrentamiento por clústers	arab	chin	eng	fran	slav	iber	Media Clúster
exp_lang_1	91.1578	90.9814	93.8111	100	91.4634	91.2052	93.10315
exp_lang_2	93.6914	93.5013	95.114	100	94.7154	93.1596	95.03028333
exp_lang_3	95.6929	95.9549	96.9598	100	97.9675	94.6254	96.86675
exp_lang_4	95.7436	96.0875	97.0141	100	98.7805	94.2997	96.98756667

TABLA 5.6: Índices de acierto de cada clúster de idioma y media de clústers para cada sistema de reconocimiento de idioma que usa características BN.

La anterior tabla 5.6 muestra, por cada experimento distinto, los índices de acierto de los seis clústers en los que se dividen todos los segmentos y la media de todos ellos. Esta es una media aritmética de los índices de acierto sin ponderar por el número de segmentos que tiene cada clúster.

Algo llamativo es el acierto de absolutamente todos los segmentos del clúster francés, que termina de perfeccionar el ya de por si alto porcentaje de acierto del experimento *exp_mfcc* de 97.2222%. Esto era esperable ya que resultados de la evaluación realizada en el grupo Audias-ATVS con otro tipo de redes (*Long Short-Term Memory*, LSTM) también dieron resultados anómalos relacionados con este clúster. Los organizadores de NIST y otros participantes detectaron anomalías como que los segmentos de cada idioma provenían de distintos canales, pudiendo así sobreentrenarse y aprender características no asociadas a la tarea de reconocimiento de idioma como por ejemplo esta información del canal, no consiguiendo por tanto eliminar esta variabilidad con *Total Varability*, haciendo que el sistema aprenda a clasificar canales, en lugar de idiomas.

Obviando el clúster francés que se sale de la norma general, en el resto de clústers se puede observar que todos tienen unos índices de aciertos similares y que a medida que la posición del *bottleneck* avanza en las capas estos índices aumentan, siendo el último experimento *exp_lang_4* con el *bottleneck* en la cuarta y última posición el que mejores resultados obtiene.

Así pues, usando como medida de evaluación el porcentaje de segmentos clasificados correctamente, la posición óptima del *bottleneck* será la cuarta posición porque aporta una mayor mejora respecto al experimento base. Las dos siguientes tablas (5.7 y 5.8) resumen los resultados obtenidos en *exp_mfcc* y *exp_lang_4*, así como la mejoría final conseguida tras usar satisfactoriamente las características BN extraídas del entrenamiento de una red neuronal para la tarea de clasificación de trifonemas como entrada a un sistema de reconocimiento de idioma.

Enfrentamiento total	20vs20	I.A - A	I.A - B
exp_mfcc	83.2744	81.4375	58.243
exp_lang_4	92.3261	89.8139	68.2996
Mejora (%)	9.0517	8.3764	10.0566

TABLA 5.7: Comparativa de los enfrentamientos totales para los experimentos exp_mfcc y exp_lang_4.

Enfrentamiento por clústers	arab	chin	eng	fran	slav	iber	Media Clúster
exp_mfcc	90.534	90.5235	93.7636	97.2222	96.206	87.4797	92.6215
exp_lang_4	95.7436	96.0875	97.0141	100	98.7805	94.2997	96.98756667
Mejora (%)	5.2096	5.564	3.2505	2.7778	2.5745	6.82	4.366066667

TABLA 5.8: Comparativa de los enfrentamientos de clúster y su media para los experimentos exp_mfcc y exp_lang_4.

5.4 Resultados en EER

Como ya se adelantó en el apartado *4.1.3 Modelados y evaluación de resultados*, también se ha evaluado el sistema obteniendo el EER en los enfrentamientos de los segmentos a los idiomas pertenecientes a su clúster, marcando como *Target* el idioma del segmento, y *Non-Target* los otros idiomas pertenecientes al clúster. La tabla 5.9 muestra el EER de los

clústers en los cuatro experimentos de la etapa UBM/i-vector BN y el experimento base de la etapa UBM/i-vector MFCC, así como la media de cada experimento. Esta media, al igual que el resto de medias relacionadas con resultados de clústers son medias sin ponderar por el número de segmentos de cada clúster. El EER suele ser la medida estándar de rendimiento en sistemas de reconocimiento tanto de idioma como de locutor [19] por lo que se hace necesario comentar estos resultados.

Experimento	exp_mfcc	exp_lang_1	exp_lang_2	exp_lang_3	exp_lang_4
media EER	7.3455	8.9568	6.7017	4.364	4.8721
arab	4.9748	6.2077	4.9277	2.5738	2.2453
chin	7.2415	8.5043	5.6284	4.1923	3.8645
eng	2.0323	5.8967	3.8633	1.9054	2.287
fren	6.0417	2.5	0	1.6667	3.3333
slav	9.1005	17.4147	12.301	5.4349	5.0345
iber	14.6824	13.2173	13.4895	10.4106	12.4681

TABLA 5.9: EER por clúster de idioma y la media de estos para el experimento base y los cuatro experimentos de idioma con características BN.

Pasando por alto los resultados del clúster francés por motivos ya comentados anteriormente, se puede apreciar cómo la rebaja de la media del EER no se consigue con el primer experimento, y en algunos clústers esta no se consigue hasta el tercer experimento. Esto quiere decir que pese a que en términos de precisión en la correcta identificación del idioma de un segmento se incremente usando características BN extraídas de un *bottleneck* en cualquier capa oculta, esto no implica que se mejore los resultados en términos de EER con el *bottleneck* en cualquier posición. El gráfico de la figura 5.1 muestra los resultados de la anterior tabla 5.9 en un diagrama de barras para visualizar mejor la evolución del EER según la posición del *bottleneck* en la red.

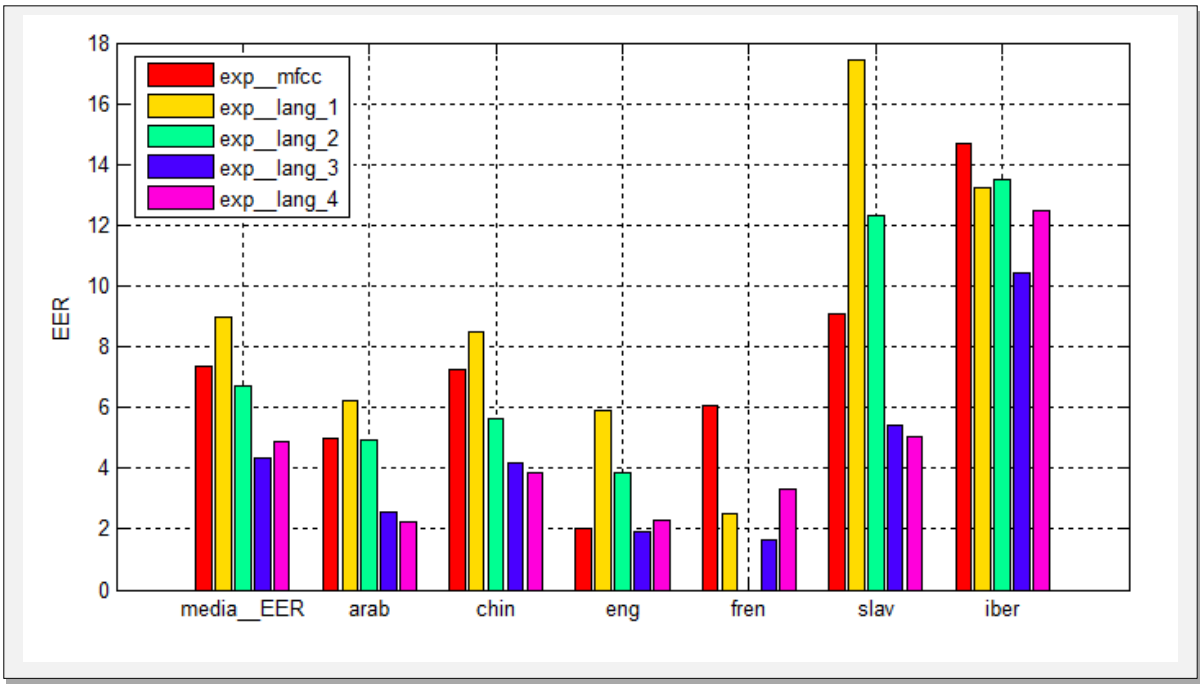

FIGURA 5.1: EER para cada clúster y la media de estos según el experimento.

Se observa cómo el mínimo EER converge en los experimentos en los que se ha usado características de un *bottleneck* en las últimas capas ocultas, habiendo una diferencia bastante notable entre los dos primeros experimentos y los dos últimos, siendo en términos generales *exp_lang_3*, con el *bottleneck* en la tercera posición sobre cuatro, el experimento con un menor EER.

Esto implica que aunque con nuestro objetivo de dar una clasificación de idioma, eligiendo aquel que proporcione la puntuación más alta, sea la posición óptima del *bottleneck* en la última capa oculta, no tiene por qué ser así para todos los sistemas. Hay que ser consciente que esta posición no será siempre la idónea cuando se busque un sistema que tenga en cuenta algo más que la tasa de acierto, como por ejemplo que la Falsa Aceptación y el Falso Rechazo tengan un tratamiento distinto.

6. CONCLUSIONES Y TRABAJO FUTURO

Completados los experimentos y una vez vistos los resultados, estamos en posición de responder a las preguntas que se planteaban al comienzo de este Trabajo de Fin de Grado.

Se ha comprobado y medido la mejora que supone el uso de características BN extraídas de una red neuronal profunda entrenada para la tarea de clasificación de trifonemas, en sustitución de características acústicas MFCC como entrada a un sistema de reconocimiento de idioma. Para la identificación de idioma en enfrenamientos de segmentos restringidos por clústers se consigue una mejora de un 4.37% de media, y de hasta 6.82% para clusters específicos, que partiendo de una tasa de acierto del 92.62% supone una mejora muy a tener en cuenta. En una clasificación tras un enfrentamiento total con cualquiera de los veinte idiomas posibles, donde el margen de mejora era superior, también ha sido superior la mejora obtenida incrementando el número de segmentos acertados en un 9.05%.

Aparte de comprobar la mejora que pudiese suponer en un sistema que ya parte de un índice de acierto muy alto, otro objetivo de este trabajo era observar cómo afecta la posición del *bottleneck* en las características extraídas de la red neuronal como entrada al sistema de reconocimiento de idioma.

Partiendo de la premisa de que una capa oculta de la red neuronal tendrá información más relevante respecto al objetivo de su entrenamiento según se vaya acercando de la capa de entrada a la de salida, se ha comprobado cómo el acercamiento del *bottleneck* a la capa de salida de la red neuronal proporciona unas características de mayor utilidad para nuestro sistema de reconocimiento de idioma, incrementando el índice de acierto situando el *bottleneck* en cualquiera de las cuatro capas ocultas y siendo la posición óptima para esta tarea la cuarta y última capa.

No obstante, un análisis del EER obtenido para los distintos experimentos nos hace ver cómo no hay una relación directa entre este aumento lineal de la mejora (en términos de índice de acierto) de nuestro sistema base con el EER alcanzado. Este análisis revela que es la tercera capa y no la cuarta donde la posición del *bottleneck* proporciona unas características óptimas en términos de EER, bajándolo de un 7.35% a un 4.36% de media. También revela que esta mejora no se consigue con el *bottleneck* en cualquier posición, ya que colocado en la primera capa oculta el EER aumenta con respecto al experimento base, y en la segunda capa sólo hay mejoras destacables en dos de los seis clústers.

Por lo que usando como medida de rendimiento el EER, concluimos en primer lugar que no siempre el uso de características BN en sustitución a características acústicas MFCC como parámetros de entrada supondrán una mejora al sistema base, y en segundo lugar que el EER no descenderá linealmente a medida que el *bottleneck* se acerque a la capa de salida de la red neuronal profunda.

Existe por tanto un compromiso entre características BN extraídas de una capa cercana a la entrada de la red neuronal donde no serán suficientemente discriminativas, y las características extraídas de un *bottleneck* cercanas a la salida de la red, donde el entrenamiento para una tarea distinta tenga demasiado peso y afecte negativamente a la utilidad de esas características para la tarea buscada.

Hemos probado en este Trabajo Fin de Grado cómo la sustitución de características acústicas MFCC por características BN de las últimas capas de una red neuronal con la tarea de clasificación de trifonemas proporciona una mejora significativa en un sistema de identificación de idioma, tanto en términos de índice de acierto como de EER. Sería interesante comprobar la relación entre distintas tareas para las que se puede entrenar una red, y tareas de sistemas que se pretenden mejorar con el cambio de características, así como distintos niveles de entrenamiento alcanzado en las redes neuronales.

Para combinaciones en las que una tarea tienda a ignorar información relevante para la consecución de la otra, se ha probado que a partir de las características BN extraídas de redes neuronales con un entrenamiento subóptimo se llegan a mejores resultados que usando características de una red con un entrenamiento completo [27], como en el caso de redes entrenadas para reconocimiento del habla, donde un entrenamiento completo hace que las características que se puedan extraer omitan información de locutor, siendo estas por tanto no convenientes como entrada a un sistema de reconocimiento de locutor.

Aunque para tareas con objetivos divergentes no se pueda saber con anterioridad a la realización de experimentos qué nivel de entrenamiento y qué posición del *bottleneck* son los apropiados para conseguir las mejores características posibles, sí que se espera que para tareas relacionadas, o incluso la misma tarea, una red plenamente entrenada y un *bottleneck* en las últimas capas ocultas proporcionen unas características idóneas.

6.1 Aportaciones al trabajo

En este último apartado de la memoria, se desglosará qué material fue aportado por la tutora, Alicia Lozano Díez, y qué se realizó durante el desarrollo de este Trabajo Fin de Grado mientras se sigue una vista general del planteamiento llevado a cabo a modo de resumen.

Para el sistema de reconocimiento de idioma UBM/i-vector se proporcionaron los scripts pertenecientes sistema original que usa coeficientes MFCC y el script usado para su evaluación EER. Durante la primera etapa del desarrollo, referida a la implementación de nuestro experimento base *exp_mfcc*, se realizaron todas las fases implicadas en su entrenamiento y evaluación, incluyendo la extracción de características de las locuciones LRE15. A lo largo del proceso se modificaron los scripts proporcionados amoldándolos al entorno de trabajo, y tras el entrenamiento y obtención de los primeros resultados de evaluación se diseñaron y probaron las implementaciones alternativas. La realización de todas las etapas de este primer experimento fue básica no sólo para habituarse al entorno y a la herramienta Kaldi (que no había sido usada en ninguna asignatura del Grado), además sirvió como toma de contacto para la medición de tiempos y recursos usados en cada fase de entrenamiento de cara una a mejor organización de los experimentos UBM/i-vector BN.

En el experimento DNN-BN, las características usadas para el entrenamiento del sistema y para la extracción de las características BN, así como sus etiquetas asociadas, fueron proporcionadas junto con los scripts del experimento original, centrándonos durante esta segunda etapa del desarrollo principalmente en la gestión del entorno para un correcto aprovechamiento de tiempo, recursos y espacio. La modificación de los scripts proporcionados requirió de nuevo el aprendizaje de otra herramienta con la que no tenía

experiencia, Theano, y también exigió el correcto uso del entorno GPU, con el que no había tenido oportunidad de trabajar hasta la fecha.

En la última etapa del trabajo, UBM/i-vector BN, se volvieron a modificar los scripts del experimento base adaptando todo el sistema a las nuevas características de entrada BN y una vez completados todos los entrenamientos y evaluaciones se tuvo lo necesario para llegar a los resultados y conclusiones mostrados en esta memoria.

REFERENCIAS

[1] John H.L. Hansen & Taufiq Hasan. IEEE Signal Processing Magacine [99] (November 2015). *Speaker Recognition by Machines and Humans (Fig.6.c, p. 85).*

[2] J. Benesty, M. Sondhi, & Y. Huang. Eds (2008). *Springer handbook of speech processing, Chapter 41.2.1-Shifted Delta Cepstral Features (Fig.41.1, p. 812).*

[3] J. Benesty, M. Sondhi, & Y. Huang. Eds (2008). *Springer handbook of speech processing, Chapter 41.2.2-Classifiers (Fig.41.3, p. 814).*

[4] Pedro A. Torres-Carrasquillo, Elliot Singer, Mary A. Kohler, Richard J. Greene, Douglas A. Reynolds & J.R. Deller, Jr. 7th International Conference on Spoken Language Processing (September 2002, Denver, Colorado, USA). *Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features (Fig.2, p.90).*

[5] J. Benesty, M. Sondhi, & Y. Huang. Eds (2008). *Springer handbook of speech processing, Chapter 39.2 – Language Recognition Principles (pp. 786-788).*

[6] Geoffrey Hinton, Li Deng, Dong Yu & George E. Dahl. IEEE Signal Processing Magazine, vol.29, Issue.6, (November 2012). *Deep Neural Networks for Acoustic Modelling in Speech Recognition: The Shared Views of Four Research Groups*.

[7] J. Benesty, M. Sondhi, & Y. Huang. Eds (2008). *Springer handbook of speech processing, Chapter 39.1-Spoken Language (pp. 785-786).*

[8] M. Zissman & E. Singer. Proceedings of the International Conference on Acoustics Speech and Signal Processing (1994). *Automatic Language Identification of Telephone Speech Messages Using Phoneme Recognition and n -Gram Modeling (pp. 305–308).*

[9] Y. Yan & E. Barnard. Proceedings of the International Conference on Acoustics Speech and Signal Processing (1995). *An Approach to Automatic Language Identification Based on Language-Dependent Phone Recognition (pp. 3511–3514).*

[10] M.A. Zissman & K.M. Berkling. Automatic language identification (2001). *Speech Communication, 35 (pp. 115– 124).*

[11] John H.L. Hansen & Taufiq Hasan. IEEE Signal Processing Magacine [99] (November 2015). *Speaker Recognition by Machines and Humans- Short-Term Features (pp. 84-85).*

[12] S. Davis & P. Mermelstein. IEEE Trans. Acoust. Speech Signal Processing, vol.28, no.4, (1980). *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences (pp. 357–366).*

[13] Patrick Kenny. Draft version (January 2006). *Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms.*

[14] Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, & Pierre Ouellet. IEEE Transactions on audio, speech, and language processing, vol.19, no4, (May 2011). *Front-End Factor Analysis for Speaker Verification.*

[15] Santiago Fernandez, Alex Graves & Jürgen Schmidhuber. Proceedings of ICANN (2) (2007). *An application of recurrent neural networks to discriminative keyword spotting (pp. 220–229).*

[16] Pavel Matějka, Le Zhang, Tim Ng, Sri Harish Mallidi, Ondřej Glembek, Jeff Ma & Bing Zhang . The Speaker and Language Recognition Workshop (June 2014, Joensuu, Finland). *Neural Network Bottleneck Features for Language Identification.*

[17] Daniel Povey, Arnab Ghoshal,Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer & Karel Veselý. *The Kaldi Speech Recognition Toolkit.*

[18] The Theano Development Team. *Theano: A Python framework for fast computation of mathematical expressions.*

[19] NIST (2015). *The 2015 NIST Language Recognition Evaluation Plan.*

[20] *https://catalog.ldc.upenn.edu/docs/LDC97S62/swb1_manual.txt. (Checked may 2017)*

[21] *https://catalog.ldc.upenn.edu/docs/LDC97S62/release_2_table_updates.txt (Checked may 2017)*

[22] Pedro A. Torres-Carrasquillo, Elliot Singer, Mary A. Kohler, Richard J. Greene, Douglas A. Reynolds & J.R. Deller, Jr. 7th International Conference on Spoken Language Processing (September 2002, Denver, Colorado, USA). *Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features (pp. 89-92).*

[23] Jonathan Kola, Carol Espy-Wilson & Tarun Pruthi. MERIT BIEN (2011). *Voice Activity Detection.*

[24] Van Segbroeck, Maarten, Andreas Tsiartas & Shrikanth Narayanan. INTERSPEECH (2013). *A robust frontend for VAD: exploiting contextual, discriminative and spectral cues of human voice.*

[25] Amit Singhal. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4), (2001). *Modern Information Retrieval: A Brief Overview (pp. 35–43).*

[26] Tom Fawcett. Pattern Recognition Letters 27 (2006). *An introduction to ROC analysis (pp. 861–874).*

[27] Alicia Lozano-Diez, Anna Silnova, Pavel Matějka, Ondřej Glembek, Oldřich Plchot, Jan Pešán, Lukáš Burget & Joaquin Gonzalez-Rodriguez. Odyssey 2016 (Bilbao, Spain, June 2016). *Analysis and Optimization of Bottleneck Features for Speaker Recognition.*

[28] D. Michie, D.J. Spiegelhalter & C.C. Taylor. Ellis Horwood (1994). *Machine Learning, Neural and Statistical Classification (pp. 90-105).*

GLOSARIO

ALR: Automatic Language Recognition,
API: Application Programming Interface
ASR: Automatic Speech Recognition
BN: Bottleneck
BNBS: Broadcast Narrowband Speech
CM: Confusion Matrix
CPU: Central Processing Unit
CSA: Cosine Score Algorithm
CTS: Conversational Telephone Speech
DCT: Direct Cosine Transform
DFT: Direct Fourier Transform
DNNs: Deep Neural Networks
EER: Equal Error Rate
FST: Finite-State Transducers
GMM: Gaussian Mixed Models
GPU: Graphics Processor Unit
HMM: Hidden Markov Models
LDC: Linguistic Data Consortium
LRE15: The 2015 NIST Language Recognition Evaluation Plan
LSTM: Long Short-Term Memory
MAP: Maximum A Posteriori
MFCC: Mel Frequency Cepstral Coefficients
ML: Machine Learning
NIST: National Institute of Standards and Technology
P-PRLM: Parallel Phone Recognition followed by Language Modelling
PRLM: Phone Recognition followed by Language Modelling
RATS: Robust Automated Transcription of Speech
SDC: Shifted Delta Cepstral Features
SVM: Suport Vector Machine
T: Total Variability Space
TV: Total Variability
UBM: Universal Background Model
VAD: Voice Activity Detection

ANEXO

En este anexo se añadirá a la memoria las matrices de confusión correspondientes a la clasificación de los segmentos en idiomas para las distintas evaluaciones realizadas en los cinco experimentos del sistema de reconocimiento de idioma UBM/i-vector.

Matrices de confusión *exp_mfcc*

exp_mfcc-ARA	1	2	3	4	5
ara-arz (1)	1597	11	38	7	43
ara-acm (2)	5	671	24	7	31
ara-apc (3)	15	44	693	7	83
ara-ary (4)	3	6	11	555	39
ara-arb (5)	0	0	0	0	61

exp_mfcc-ZHO	6	7	8	9
zho-yue (6)	48	0	0	0
zho-cmn (7)	36	1186	30	54
zho-cdo (8)	8	1	62	10
zho-wuu (9)	2	0	2	70

exp_mfcc-ENG	10	11	12
eng-gbr (10)	7	0	0
eng-usg (11)	55	1598	58
eng-sas (12)	2	0	124

exp_mfcc-FRE	13	14
fre-waf (13)	57	3
fre-hat (14)	0	48

exp_mfcc -QSL	15	16
qsl-pol (15)	459	15
qsl-rus (16)	13	251

exp_mfcc-SPA	17	18	19	20
spa-car (17)	438	10	10	3
spa-eur (18)	2	72	16	0
spa-lac (19)	4	30	21	1
por-brz (20)	0	0	1	7

Matrices de confusión de las evaluaciones por clúster para el experimento *exp_mfcc*.

exp_mfcc 20VS20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1417	5	25	2	34	12	1	12	12	20	1	20	19	27	3	5	22	14	13	32
ara-acm (2)	1	608	15	4	28	26	0	6	2	6	1	5	22	0	1	1	1	5	4	2
ara-apc (3)	5	11	620	4	64	16	0	6	3	8	0	5	35	4	4	4	4	23	17	9
ara-ary (4)	0	2	6	500	31	5	1	7	7	4	0	2	10	6	2	5	2	8	9	7
ara-arb (5)	0	0	0	0	60	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
zho-yue (6)	0	0	0	0	0	46	0	0	0	0	0	0	1	0	0	0	0	1	0	0
zho-cmn (7)	3	0	1	1	7	30	1116	26	51	8	1	6	17	10	1	2	1	8	4	13
zho-cdo (8)	0	0	0	0	0	5	0	59	6	0	0	0	11	0	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	1	0	1	67	0	0	0	2	0	0	0	1	1	0	1
eng-gbr (10)	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	1
eng-usg (11)	2	0	2	1	11	16	5	21	12	38	1418	33	13	25	15	8	13	23	16	39
eng-sas (12)	0	0	0	0	0	3	0	3	1	0	0	109	2	0	0	0	0	0	8	0
fre-waf (13)	0	0	0	0	0	5	0	1	0	1	0	0	49	2	0	0	0	0	2	0
fre-hat (14)	0	0	0	0	0	1	0	0	0	0	0	0	0	47	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	9	4	1	0	5	7	0	4	7	4	419	8	0	2	2	2
qsl-rus (16)	0	0	0	1	4	1	0	1	0	5	0	4	2	3	8	228	1	1	2	3
spa-car (17)	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	0	434	9	10	1
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	72	16	0
spa-lac (19)	0	0	0	0	0	2	0	2	0	0	0	1	5	0	0	1	3	25	17	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	7

Matriz de confusión de la evaluación 20vs20 para el experimento *exp_mfcc*.

exp_mfcc Imp.Alt.A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1289	5	24	3	31	4	5	1	5	9	14	11	50	89	32	21	66	13	13	11
ara-acm (2)	3	587	19	3	26	1	3	1	2	0	2	1	37	17	4	10	8	5	7	2
ara-apc (3)	14	29	584	2	67	1	1	2	2	1	2	0	57	37	13	7	8	5	8	2
ara-ary (4)	1	5	5	439	27	2	2	2	2	0	2	5	28	31	19	18	5	10	3	8
ara-arb (5)	0	0	0	0	36	0	0	0	0	0	0	0	3	15	1	0	1	3	2	0
zho-yue (6)	0	0	0	0	0	37	0	0	0	2	0	0	8	0	0	0	0	1	0	0
zho-cmn (7)	2	0	0	0	0	32	1159	28	51	2	2	0	11	14	1	1	3	0	0	0
zho-cdo (8)	0	0	0	0	0	3	0	35	0	0	0	2	36	2	0	0	0	1	2	0
zho-wuu (9)	0	0	0	0	0	1	0	1	58	0	0	0	11	0	0	0	2	1	0	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	3	0	0	1	3	0	0	0	0	0	0
eng-usg (11)	0	0	0	0	0	1	8	1	1	42	1549	48	6	15	16	6	11	4	2	1
eng-sas (12)	0	0	0	0	0	0	0	0	1	1	0	62	19	24	7	5	0	1	5	1
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	54	3	0	0	0	0	3	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	1	0	1	0	0	0	0	0	4	6	447	13	1	0	0	1
qsl-rus (16)	0	0	0	0	0	0	1	0	1	1	0	0	8	8	13	230	0	0	1	1
spa-car (17)	0	0	1	0	0	0	0	0	0	0	0	0	1	2	0	0	435	10	10	2
spa-eur (18)	0	0	1	0	0	0	0	0	1	0	0	0	2	0	0	0	2	68	16	0
spa-lac (19)	0	1	0	0	0	0	0	0	0	0	0	0	8	5	0	1	4	22	15	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	1	3

Matriz de confusión de la evaluación de la implementación alternativa A para el experimento *exp_mfcc*.

exp_mfcc Imp.Alt.B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	701	3	17	2	36	53	1	29	23	141	9	77	124	141	56	80	35	48	49	71
ara-acm (2)	1	471	14	1	28	41	0	15	9	23	1	7	50	11	5	19	1	11	12	18
ara-apc (3)	8	23	467	1	75	27	3	15	15	20	1	20	65	39	10	9	5	10	19	10
ara-ary (4)	0	4	5	378	35	10	1	11	9	18	0	8	35	15	13	26	1	13	14	18
ara-arb (5)	0	0	0	0	59	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
zho-yue (6)	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	0	2	1	13	31	692	26	46	73	3	28	102	110	30	33	17	38	20	40
zho-cdo (8)	0	0	0	0	0	7	0	57	3	0	0	0	13	1	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	2	0	1	62	0	0	0	6	0	0	0	0	1	1	1
eng-gbr (10)	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	0	2	4	4	24	88	6	61	54	50	823	40	117	95	76	92	23	49	37	66
eng-sas (12)	0	0	0	0	0	2	0	5	2	1	0	100	6	4	1	2	0	0	3	0
fre-waf (13)	0	0	0	0	0	7	0	2	0	1	0	0	48	2	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	1	0	0	0	0	0	0	0	45	0	0	0	2	0	0
qsl-pol (15)	0	0	0	0	7	2	1	3	3	7	0	2	4	4	424	14	0	1	0	2
qsl-rus (16)	0	0	0	1	1	1	0	1	1	2	0	2	7	5	10	229	0	0	2	2
spa-car (17)	0	0	1	0	0	9	0	2	2	1	1	2	5	7	4	5	402	9	10	1
spa-eur (18)	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	2	69	16	0
spa-lac (19)	0	0	0	0	0	1	0	0	0	0	0	0	4	1	0	1	4	27	17	1
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	6

Matriz de confusión de la evaluación de la implementación alternativa B para el experimento *exp_mfcc*.

exp_mfcc imp.A-20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-128	0	-1	1	-3	-8	4	-11	-7	-11	13	-9	31	62	29	16	44	-1	0	-21
ara-acm (2)	2	-21	4	-1	-2	-25	3	-5	0	-6	1	-4	15	17	3	9	7	0	3	0
ara-apc (3)	9	18	-36	-2	3	-15	1	-4	-1	-7	2	-5	22	33	9	3	4	-18	-9	-7
ara-ary (4)	1	3	-1	-61	-4	-3	1	-5	-5	-4	2	3	18	25	17	13	3	2	-6	1
ara-arb (5)	0	0	0	0	-24	-1	0	0	0	0	0	0	3	15	1	0	1	3	2	0
zho-yue (6)	0	0	0	0	0	-9	0	0	0	2	0	0	7	0	0	0	0	0	0	0
zho-cmn (7)	-1	0	-1	-1	-7	2	43	2	0	-6	1	-6	-6	4	0	-1	2	-8	-4	-13
zho-cdo (8)	0	0	0	0	0	-2	0	-24	-6	0	0	2	25	2	0	0	0	1	2	0
zho-wuu (9)	0	0	0	0	0	0	0	0	-9	0	0	0	9	0	0	0	1	0	0	-1
eng-gbr (10)	0	0	0	0	0	0	0	0	0	-3	0	0	1	3	0	0	0	0	0	-1
eng-usg (11)	-2	0	-2	-1	-11	-15	3	-20	-11	4	131	15	-7	-10	1	-2	-2	-19	-14	-38
eng-sas (12)	0	0	0	0	0	-3	0	-3	0	1	0	-47	17	24	7	5	0	1	-3	1
fre-waf (13)	0	0	0	0	0	-5	0	-1	0	-1	0	0	5	1	0	0	0	0	1	0
fre-hat (14)	0	0	0	0	0	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	-8	-4	0	0	-5	-7	0	-4	-3	2	28	5	1	-2	-2	-1
qsl-rus (16)	0	0	0	-1	-4	-1	1	-1	1	-4	0	-4	6	5	5	2	-1	-1	-1	-2
spa-car (17)	0	0	0	0	-1	-1	0	-1	0	-1	0	-1	1	1	0	0	1	1	0	1
spa-eur (18)	0	0	1	0	0	0	0	0	1	0	0	0	2	0	0	0	0	-4	0	0
spa-lac (19)	0	1	0	0	0	-2	0	-2	0	0	0	-1	3	5	0	0	1	-3	-2	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	-4

Resta de la matriz de confusión 20vs20 a la matriz de confusión de la Implementación alternativa A para el experimento *exp_mfcc*.

Matrices de confusión *exp_lang_1*

exp_lang_1-ARA	1	2	3	4	5
ara-arz (1)	1648	0	10	19	18
ara-acm (2)	1	651	58	11	16
ara-apc (3)	3	113	699	18	9
ara-ary (4)	4	28	28	542	10
ara-arb (5)	3	0	0	0	58

exp_lang_1-ZHO	6	7	8	9
zho-yue (6)	36	0	4	8
zho-cmn (7)	34	1218	28	25
zho-cdo (8)	12	0	58	11
zho-wuu (9)	5	0	9	60

exp_lang_1-ENG	10	11	12
eng-gbr (10)	7	0	0
eng-usg (11)	42	1605	62
eng-sas (12)	4	6	116

exp_lang_1-FRE	13	14
fre-waf (13)	60	0
fre-hat (14)	0	48

exp_lang_1-QSL	15	16
qsl-pol (15)	427	47
qsl-rus (16)	16	248

exp_lang_1-SPA	17	18	19	20
spa-car (17)	447	1	10	2
spa-eur (18)	4	74	12	0
spa-lac (19)	3	22	31	0
por-brz (20)	0	0	0	8

Matrices de confusión de las evaluaciones por clúster para el experimento *exp_lang_1*.

exp_lang_1 20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1486	0	6	4	12	1	5	4	7	6	14	10	11	55	3	0	34	7	8	22
ara-acm (2)	0	568	40	2	12	21	0	4	7	9	0	6	30	4	0	10	0	10	13	1
ara-apc (3)	1	52	646	3	5	1	0	10	2	3	0	12	49	5	0	7	0	14	23	9
ara-ary (4)	1	12	12	469	3	1	0	17	18	13	0	9	8	3	2	7	0	20	8	9
ara-arb (5)	2	0	0	0	55	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1
zho-yue (6)	0	0	0	0	0	34	0	4	6	1	0	0	2	0	0	0	0	0	1	0
zho-cmn (7)	1	0	1	1	3	29	1156	21	21	12	3	8	8	14	1	1	6	11	2	6
zho-cdo (8)	0	2	0	0	0	7	0	51	6	0	0	0	15	0	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	4	0	9	57	0	0	0	3	0	0	0	0	0	1	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	2
eng-usg (11)	10	1	0	2	2	5	5	38	3	30	1444	48	16	34	2	12	21	10	6	20
eng-sas (12)	0	1	6	0	1	2	0	2	2	0	0	92	1	0	3	0	0	0	16	0
fre-waf (13)	0	0	0	0	0	4	0	2	0	0	0	0	54	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	1	1	26	1	0	0	11	3	0	10	2	0	383	29	0	6	1	0
qsl-rus (16)	0	0	1	3	19	0	0	1	1	7	0	13	2	1	11	186	1	4	9	5
spa-car (17)	0	0	0	0	0	1	1	0	0	4	0	2	0	9	0	1	433	0	9	0
spa-eur (18)	0	0	0	0	0	0	0	3	0	0	0	1	1	0	0	0	2	72	11	0
spa-lac (19)	0	1	0	0	0	7	0	0	0	0	0	0	7	0	0	3	3	9	26	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

Matriz de confusión de la evaluación 20vs20 para el experimento *exp_lang_1*.

exp_lang_1 Imp.Alt. A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1186	0	8	5	11	6	30	5	2	7	76	13	27	105	30	14	121	18	19	12
ara-acm (2)	1	538	46	6	6	2	0	2	2	2	1	1	40	26	17	29	6	5	6	1
ara-apc (3)	2	79	626	15	7	1	2	0	1	0	2	0	42	21	17	14	1	8	2	2
ara-ary (4)	3	14	16	410	3	2	1	0	1	3	0	2	34	29	30	48	3	2	10	1
ara-arb (5)	3	0	0	0	13	0	0	0	0	0	0	0	3	38	0	1	1	1	1	0
zho-yue (6)	0	0	0	0	0	14	0	4	7	0	0	0	4	8	1	0	1	2	7	0
zho-cmn (7)	0	0	1	1	0	34	1188	22	24	0	6	2	4	11	1	3	7	0	0	1
zho-cdo (8)	0	2	0	0	0	8	0	35	1	0	0	0	24	0	2	1	0	0	8	0
zho-wuu (9)	0	3	0	0	0	2	0	8	51	0	0	0	4	0	1	2	1	0	2	0
eng-gbr (10)	0	0	0	0	0	0	0	1	0	1	0	0	1	2	2	0	0	0	0	0
eng-usg (11)	2	0	0	0	0	1	8	12	0	34	1524	54	3	33	5	7	18	6	1	1
eng-sas (12)	0	0	1	0	0	0	0	0	1	0	2	46	23	17	16	6	2	1	8	3
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	56	0	2	2	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	2	0	0	0	0	0	1	0	0	0	1	5	419	43	0	1	0	2
qsl-rus (16)	0	2	3	1	0	0	0	0	0	0	0	2	7	12	15	215	4	2	1	0
spa-car (17)	0	0	0	0	0	0	1	0	0	0	0	0	0	9	1	0	437	0	10	2
spa-eur (18)	0	0	1	1	0	0	0	0	0	0	0	0	1	0	4	1	4	67	11	0
spa-lac (19)	0	2	0	0	0	0	0	0	0	0	0	0	4	0	0	1	3	17	29	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0

Matriz de confusión de la evaluación de la implementación alternativa A para el experimento *exp_lang_1*.

exp_lang_1 Imp.Alt.B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	641	0	2	3	15	30	24	23	30	95	37	54	136	230	73	83	80	40	35	64
ara-acm (2)	0	434	32	5	14	45	0	34	8	19	1	12	36	15	19	40	0	14	7	2
ara-apc (3)	1	56	532	13	9	16	1	17	13	23	0	6	48	8	13	16	1	24	22	23
ara-ary (4)	1	5	11	318	4	11	0	11	21	46	0	24	25	28	14	48	1	17	17	10
ara-arb (5)	3	0	0	0	54	0	0	0	0	3	0	0	0	0	0	0	0	1	0	0
zho-yue (6)	0	0	0	0	0	36	0	4	7	1	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	0	4	2	15	32	722	19	20	74	10	21	77	121	58	52	22	13	9	33
zho-cdo (8)	0	1	0	0	0	11	0	53	6	2	0	0	7	0	0	0	0	1	0	0
zho-wuu (9)	0	2	0	0	0	4	0	9	58	0	0	0	1	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	6	0	0	2	3	54	10	95	52	39	750	37	118	194	71	88	41	36	43	70
eng-sas (12)	0	1	0	0	2	1	0	1	6	1	0	76	11	4	9	2	0	0	9	3
fre-waf (13)	0	0	0	0	0	2	0	1	1	0	0	0	53	0	1	2	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	5	2	0	0	2	10	0	3	1	3	404	41	0	2	0	1
qsl-rus (16)	0	1	1	0	1	0	0	1	0	8	0	10	3	5	12	213	1	2	2	4
spa-car (17)	0	0	0	0	1	7	0	6	1	22	0	6	51	25	11	9	313	0	7	1
spa-eur (18)	0	0	0	0	0	1	0	4	1	1	0	0	0	0	5	1	3	65	9	0
spa-lac (19)	0	1	0	0	0	1	0	2	1	1	0	0	13	6	0	0	3	16	12	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

Matriz de confusión de la evaluación de la implementación alternativa B para el experimento *exp_lang_1*.

exp_lang_1 imp.A-20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-300	0	2	1	-1	5	25	1	-5	1	62	3	16	50	27	14	87	11	11	-10
ara-acm (2)	1	-30	6	4	-6	-19	0	-2	-5	-7	1	-5	10	22	17	19	6	-5	-7	0
ara-apc (3)	1	27	-20	12	2	0	2	-10	-1	-3	2	-12	-7	16	17	7	1	-6	-21	-7
ara-ary (4)	2	2	4	-59	0	1	1	-17	-17	-10	0	-7	26	26	28	41	3	-18	2	-8
ara-arb (5)	1	0	0	0	-42	0	0	0	0	-1	0	-1	3	38	0	1	1	1	0	-1
zho-yue (6)	0	0	0	0	0	-20	0	0	1	-1	0	0	2	8	1	0	1	2	6	0
zho-cmn (7)	-1	0	0	0	-3	5	32	1	3	-12	3	-6	-4	-3	0	2	1	-11	-2	-5
zho-cdo (8)	0	0	0	0	0	1	0	-16	-5	0	0	0	9	0	2	1	0	0	8	0
zho-wuu (9)	0	3	0	0	0	-2	0	-1	-6	0	0	0	1	0	1	2	1	0	1	0
eng-gbr (10)	0	0	0	0	0	0	0	1	0	-4	0	0	1	2	2	0	0	0	0	-2
eng-usg (11)	-8	-1	0	-2	-2	-4	3	-26	-3	4	80	6	-13	-1	3	-5	-3	-4	-5	-19
eng-sas (12)	0	-1	-5	0	-1	-2	0	-2	-1	0	2	-46	22	17	13	6	2	1	-8	3
fre-waf (13)	0	0	0	0	0	-4	0	-2	0	0	0	0	2	0	2	2	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
qsl-pol (15)	0	0	1	-1	-26	-1	0	0	-10	-3	0	-10	-1	5	36	14	0	-5	-1	2
qsl-rus (16)	0	2	2	-2	-19	0	0	-1	-1	-7	0	-11	5	11	4	29	3	-2	-8	-5
spa-car (17)	0	0	0	0	0	-1	0	0	0	-4	0	-2	0	0	1	-1	4	0	1	2
spa-eur (18)	0	0	1	1	0	0	0	-3	0	0	0	-1	0	0	4	1	2	-5	0	0
spa-lac (19)	0	1	0	0	0	-7	0	0	0	0	0	0	-3	0	0	-2	0	8	3	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-8

Resta de la matriz de confusión 20vs20 a la matriz de confusión de la Implementación alternativa A para el experimento *exp_lang_1*.

Matrices de confusión *exp_lang_2*

exp_lang_2-ARA	1	2	3	4	5
ara-arz (1)	1663	0	17	4	11
ara-acm (2)	3	667	43	7	17
ara-apc (3)	4	79	727	10	22
ara-ary (4)	2	13	8	582	7
ara-arb (5)	2	0	0	0	59

exp_lang_2-ZHO	6	7	8	9
zho-yue (6)	38	0	8	2
zho-cmn (7)	6	1251	13	35
zho-cdo (8)	14	0	56	11
zho-wuu (9)	0	0	9	65

exp_lang_2-ENG	10	11	12
eng-gbr (10)	7	0	0
eng-usg (11)	14	1621	74
eng-sas (12)	2	0	124

exp_lang_2-FRE	13	14
fre-waf (13)	60	0
fre-hat (14)	0	48

exp_lang_2-QSL	15	16
qsl-pol (15)	446	28
qsl-rus (16)	11	253

exp_lang_2-SPA	17	18	19	20
spa-car (17)	450	2	8	0
spa-eur (18)	1	74	15	0
spa-lac (19)	1	11	40	4
por-brz (20)	0	0	0	8

Matrices de confusión de las evaluaciones por clúster para el experimento *exp_lang_2*.

exp_lang_2 20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1553	0	13	2	7	4	2	2	2	6	2	7	4	40	2	1	13	9	4	22
ara-acm (2)	0	596	31	2	12	20	0	1	10	3	0	0	26	12	1	1	0	8	13	1
ara-apc (3)	2	42	671	1	7	5	1	15	5	8	1	1	38	1	1	4	1	19	17	2
ara-ary (4)	1	4	4	519	2	1	0	8	6	3	0	1	19	5	4	4	0	17	5	9
ara-arb (5)	2	0	0	0	55	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0
zho-yue (6)	0	0	0	0	0	36	0	8	2	2	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	0	0	1	0	5	1212	12	31	6	2	1	3	12	1	1	2	6	1	8
zho-cdo (8)	0	1	0	0	0	10	0	53	10	0	0	0	7	0	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	0	0	8	57	0	0	0	8	0	0	0	0	0	1	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	0	0	1	0	1	1	3	20	9	12	1490	52	10	36	1	8	25	13	7	20
eng-sas (12)	0	1	8	0	0	1	0	2	4	0	0	102	1	0	0	1	0	0	5	1
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	1	0	25	0	0	0	9	3	0	5	14	0	396	15	0	4	1	1
qsl-rus (16)	0	0	1	3	8	0	0	1	1	4	0	13	4	3	2	215	0	2	4	3
spa-car (17)	0	0	0	0	0	0	1	0	0	2	0	0	1	0	0	0	446	2	8	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	74	15	0
spa-lac (19)	0	0	0	0	0	11	0	0	0	0	0	0	7	0	0	1	1	4	32	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

Matriz de confusión de la evaluación 20vs20 para el experimento *exp_lang_2*.

exp_lang_2 Imp.Alt. A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1365	0	15	2	6	2	6	0	2	2	35	3	28	91	18	13	61	12	18	16
ara-acm (2)	3	553	38	4	12	2	0	0	2	0	0	1	49	33	6	19	5	6	4	0
ara-apc (3)	2	56	678	8	10	1	1	0	0	0	1	1	46	6	10	13	2	5	1	1
ara-ary (4)	2	6	5	469	3	0	0	2	6	0	0	0	35	26	19	28	1	2	7	1
ara-arb (5)	2	0	0	0	15	0	0	0	0	0	0	0	3	39	0	0	0	0	2	0
zho-yue (6)	0	0	0	0	0	17	0	8	1	0	0	0	10	1	0	0	0	6	5	0
zho-cmn (7)	0	0	1	0	0	6	1233	12	33	0	2	1	0	10	2	1	2	0	2	0
zho-cdo (8)	0	1	0	0	0	3	0	43	1	0	0	0	24	8	0	0	0	0	1	0
zho-wuu (9)	0	1	0	0	1	0	0	7	55	0	0	0	6	0	0	0	0	0	4	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	2	0	0	1	2	2	0	0	0	0	0
eng-usg (11)	0	0	0	0	0	0	5	2	3	13	1556	58	9	42	3	0	12	2	1	3
eng-sas (12)	0	0	2	0	0	0	0	0	0	0	0	54	14	15	15	15	2	0	6	3
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	2	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	1	0	0	0	0	1	1	0	0	0	0	9	432	27	1	0	1	1
qsl-rus (16)	0	1	0	0	0	0	0	0	0	0	0	1	4	7	11	239	1	0	0	0
spa-car (17)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	449	2	8	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	73	15	0
spa-lac (19)	0	0	0	0	0	0	0	0	0	0	0	0	4	8	0	0	1	4	37	2
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0

Matriz de confusión de la evaluación de la implementación alternativa A para el experimento *exp_lang_2*.

exp_lang_2 Imp.Alt. B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	698	0	4	0	8	30	7	12	23	97	15	92	121	217	78	109	45	30	29	80
ara-acm (2)	0	466	28	3	12	53	0	16	17	6	0	6	51	31	7	20	2	12	7	0
ara-apc (3)	2	50	578	4	18	18	1	26	23	14	1	8	39	5	10	10	1	17	10	7
ara-ary (4)	1	4	3	371	4	6	1	15	13	33	0	9	40	25	12	34	0	14	14	13
ara-arb (5)	2	0	0	0	55	0	0	0	0	2	0	0	0	2	0	0	0	0	0	0
zho-yue (6)	0	0	0	0	0	38	0	8	2	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	1	0	0	18	4	810	11	33	70	1	17	84	122	45	24	9	16	10	29
zho-cdo (8)	0	0	0	0	0	14	0	49	10	5	0	0	3	0	0	0	0	0	0	0
zho-wuu (9)	0	1	0	0	1	0	0	9	63	0	0	0	0	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	6	0	0	1	0	0	0	0	0	0	0
eng-usg (11)	1	0	0	3	6	34	6	84	73	14	891	37	90	211	39	60	24	38	37	61
eng-sas (12)	0	0	1	0	0	2	0	4	2	1	0	89	2	3	4	9	0	1	6	2
fre-waf (13)	0	0	0	0	0	1	0	0	0	0	0	0	59	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	1	0	9	0	0	4	5	8	0	5	1	4	411	23	0	0	1	2
qsl-rus (16)	0	0	1	1	1	1	0	3	0	8	0	6	3	2	9	228	1	0	0	0
spa-car (17)	0	0	0	0	1	10	1	6	2	15	0	2	9	16	8	9	376	2	3	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	72	15	0
spa-lac (19)	0	0	0	0	0	4	0	4	0	0	0	0	5	0	0	8	1	10	23	1
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	6

Matriz de confusión de la evaluación de la implementación alternativa B para el experimento *ex exp_lang_2*.

exp_lang_2 imp.A-20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-188	0	2	0	-1	-2	4	-2	0	-4	33	-4	24	51	16	12	48	3	14	-6
ara-acm (2)	3	-43	7	2	0	-18	0	-1	-8	-3	0	1	23	21	5	18	5	-2	-9	-1
ara-apc (3)	0	14	7	7	3	-4	0	-15	-5	-8	0	0	8	5	9	9	1	-14	-16	-1
ara-ary (4)	1	2	1	-50	1	-1	0	-6	0	-3	0	-1	16	21	15	24	1	-15	2	-8
ara-arb (5)	0	0	0	0	-40	0	0	0	0	-1	0	0	3	38	0	0	0	-1	1	0
zho-yue (6)	0	0	0	0	0	-19	0	0	-1	-2	0	0	10	1	0	0	0	6	5	0
zho-cmn (7)	-1	0	1	-1	0	1	21	0	2	-6	0	0	-3	-2	1	0	0	-6	1	-8
zho-cdo (8)	0	0	0	0	0	-7	0	-10	-9	0	0	0	17	8	0	0	0	0	1	0
zho-wuu (9)	0	1	0	0	1	0	0	-1	-2	0	0	0	-2	0	0	0	0	0	3	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	-5	0	0	1	2	2	0	0	0	0	0
eng-usg (11)	0	0	-1	0	-1	-1	2	-18	-6	1	66	6	-1	6	2	-8	-13	-11	-6	-17
eng-sas (12)	0	-1	-6	0	0	-1	0	-2	-4	0	0	-48	13	15	15	14	2	0	1	2
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	-2	0	0	0	0	0	2	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	-25	0	0	1	-8	-3	0	-5	-14	9	36	12	1	-4	0	0
qsl-rus (16)	0	1	-1	-3	-8	0	0	-1	-1	-4	0	-12	0	4	9	24	1	-2	-4	-3
spa-car (17)	0	0	0	0	0	0	0	0	0	-2	0	0	-1	0	0	0	3	0	0	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	0	0
spa-lac (19)	0	0	0	0	0	-11	0	0	0	0	0	0	-3	8	0	-1	0	0	5	2
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-8

Resta de la matriz de confusión 20vs20 a la matriz de confusión de la Implementación alternativa A para el experimento *exp_lang_2*.

Matrices de confusión *exp_lang_3*

exp_lang_3-ARA	1	2	3	4	5
ara-arz (1)	1679	1	7	1	7
ara-acm (2)	0	698	11	1	27
ara-apc (3)	1	40	753	11	37
ara-ary (4)	0	10	6	586	10
ara-arb (5)	0	0	0	0	61

exp_lang_3-ZHO	6	7	8	9
zho-yue (6)	35	0	10	3
zho-cmn (7)	5	1275	10	15
zho-cdo (8)	7	0	69	5
zho-wuu (9)	1	0	5	68

exp_lang_3-ENG	10	11	12
eng-gbr (10)	7	0	0
eng-usg (11)	10	1655	44
eng-sas (12)	2	0	124

exp_lang_3-FRE	13	14
fre-waf (13)	60	0
fre-hat (14)	0	48

exp_lang_3-QSL	15	16
qsl-pol (15)	465	9
qsl-rus (16)	6	258

exp_lang_3-SPA	17	18	19	20
spa-car (17)	458	0	2	0
spa-eur (18)	0	78	12	0
spa-lac (19)	1	18	37	0
por-brz (20)	0	0	0	8

Matrices de confusión de las evaluaciones por clúster para el experimento *exp_lang_3*.

exp_lang_3 20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1591	1	7	0	6	1	1	0	2	4	0	4	4	35	0	1	7	3	2	26
ara-acm (2)	0	640	11	1	25	13	0	2	2	2	0	1	22	6	0	1	0	5	4	2
ara-apc (3)	1	20	701	2	23	3	0	9	1	2	0	1	42	1	0	1	0	20	15	0
ara-ary (4)	0	5	6	543	8	1	0	5	3	1	0	0	17	0	1	1	0	6	3	12
ara-arb (5)	0	0	0	0	57	0	0	0	0	1	0	0	0	2	0	0	0	1	0	0
zho-yue (6)	0	0	0	0	0	35	0	10	3	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	0	0	0	0	4	1252	9	14	3	1	2	8	3	0	0	1	1	1	5
zho-cdo (8)	0	0	0	0	0	6	0	65	5	0	0	0	4	0	0	0	0	1	0	0
zho-wuu (9)	0	1	0	0	0	1	0	4	62	0	0	0	6	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	1	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	0	1	0	0	3	2	5	17	7	9	1577	26	2	8	2	5	14	19	0	12
eng-sas (12)	0	0	6	0	0	2	0	3	2	0	0	102	1	0	0	1	0	0	8	1
fre-waf (13)	0	0	0	0	0	1	0	0	0	0	0	0	59	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	17	0	0	0	4	2	0	6	14	2	418	3	0	4	4	0
qsl-rus (16)	0	2	1	0	3	0	0	1	1	3	0	0	3	0	4	240	0	3	1	2
spa-car (17)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	456	0	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	12	0
spa-lac (19)	0	0	0	0	0	3	0	1	0	0	0	0	5	0	0	1	1	16	29	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

Matriz de confusión de la evaluación 20vs20 para el experimento *exp_lang_3*.

exp_lang_3 Imp.Alt. A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1465	1	7	1	5	0	3	1	1	1	14	3	17	97	9	5	40	7	13	5
ara-acm (2)	0	604	10	1	16	2	0	0	0	0	1	0	58	35	1	6	0	0	3	0
ara-apc (3)	1	31	703	8	27	0	0	1	2	0	0	1	46	9	6	3	0	1	3	0
ara-ary (4)	0	4	2	514	8	0	0	1	1	0	0	1	26	16	11	18	2	4	3	1
ara-arb (5)	0	0	0	0	24	0	0	0	0	0	0	0	3	32	1	0	0	0	1	0
zho-yue (6)	0	0	0	0	0	28	0	10	3	0	0	0	2	0	0	1	0	2	2	0
zho-cmn (7)	0	0	0	0	0	4	1271	9	14	0	3	0	1	2	0	0	1	0	0	0
zho-cdo (8)	0	1	0	0	0	3	0	46	1	0	0	1	21	7	0	1	0	0	0	0
zho-wuu (9)	0	0	0	0	0	0	0	4	58	0	0	0	10	0	0	0	0	0	2	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	3	0	0	0	3	1	0	0	0	0	0
eng-usg (11)	0	0	0	0	0	0	5	1	1	10	1636	39	0	8	0	3	4	0	1	1
eng-sas (12)	0	0	1	0	0	0	0	0	0	0	0	93	9	9	5	7	0	0	2	0
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	0	0	0	0	0	0	0	0	0	6	460	8	0	0	0	0
qsl-rus (16)	0	1	1	0	0	0	0	0	0	0	0	0	6	4	5	246	0	0	1	0
spa-car (17)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	457	0	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	75	12	0
spa-lac (19)	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	1	17	33	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0

Matriz de confusión de la evaluación de la implementación alternativa A para el experimento *exp_lang_3*.

exp_lang_3 Imp.Alt. B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	792	1	4	1	7	29	1	19	21	82	6	76	128	203	41	79	42	28	48	87
ara-acm (2)	0	536	10	1	24	37	0	15	8	1	0	1	47	20	2	14	0	8	4	9
ara-apc (3)	1	25	634	2	35	22	0	14	6	8	0	2	53	11	5	3	0	9	9	3
ara-ary (4)	0	3	1	408	10	5	0	9	17	17	0	5	33	19	11	31	1	11	10	21
ara-arb (5)	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
zho-yue (6)	0	0	0	0	0	35	0	10	3	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	0	0	0	1	12	4	815	10	13	75	0	18	99	114	19	32	8	18	6	61
zho-cdo (8)	0	0	0	0	0	6	0	67	5	1	0	0	1	0	0	0	0	1	0	0
zho-wuu (9)	0	0	0	0	0	1	0	5	66	0	0	0	2	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	1	1	1	1	20	55	6	63	53	10	968	25	71	110	42	85	31	36	25	105
eng-sas (12)	0	0	0	0	0	4	0	6	5	0	0	104	1	1	1	2	0	0	2	0
fre-waf (13)	0	0	0	0	0	2	0	0	0	0	0	0	58	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	5	0	0	0	0	9	0	4	1	3	438	6	0	2	5	1
qsl-rus (16)	0	1	1	0	0	0	0	1	0	3	0	0	3	1	5	248	0	0	0	1
spa-car (17)	0	0	0	0	0	4	0	3	0	4	0	0	1	5	1	10	430	0	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	77	12	0
spa-lac (19)	0	0	0	0	0	0	0	3	0	0	0	0	3	0	3	6	1	17	23	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	6

Matriz de confusión de la evaluación de la implementación alternativa B para el experimento *exp_lang_3*.

exp_lang_3 imp.A-20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-126	0	0	1	-1	-1	2	1	-1	-3	14	-1	13	62	9	4	33	4	11	-21
ara-acm (2)	0	-36	-1	0	-9	-11	0	-2	-2	-2	1	-1	36	29	1	5	0	-5	-1	-2
ara-apc (3)	0	11	2	6	4	-3	0	-8	1	-2	0	0	4	8	6	2	0	-19	-12	0
ara-ary (4)	0	-1	-4	-29	0	-1	0	-4	-2	-1	0	1	9	16	10	17	2	-2	0	-11
ara-arb (5)	0	0	0	0	-33	0	0	0	0	-1	0	0	3	30	1	0	0	-1	1	0
zho-yue (6)	0	0	0	0	0	-7	0	0	0	0	0	0	2	0	0	1	0	2	2	0
zho-cmn (7)	-1	0	0	0	0	0	19	0	0	-3	2	-2	-7	-1	0	0	0	-1	-1	-5
zho-cdo (8)	0	1	0	0	0	-3	0	-19	-4	0	0	1	17	7	0	1	0	-1	0	0
zho-wuu (9)	0	-1	0	0	0	-1	0	0	-4	0	0	0	4	0	0	0	0	0	2	0
eng-gbr (10)	0	0	0	0	-1	0	0	0	0	-3	0	0	0	3	1	0	0	0	0	0
eng-usg (11)	0	-1	0	0	-3	-2	0	-16	-6	1	59	13	-2	0	-2	-2	-10	-19	1	-11
eng-sas (12)	0	0	-5	0	0	-2	0	-3	-2	0	0	-9	8	9	5	6	0	0	-6	-1
fre-waf (13)	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	-17	0	0	0	-4	-2	0	-6	-14	4	42	5	0	-4	-4	0
qsl-rus (16)	0	-1	0	0	-3	0	0	-1	-1	-3	0	0	3	4	1	6	0	-3	0	-2
spa-car (17)	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	-3	0	0
spa-lac (19)	0	0	0	0	0	-3	0	-1	0	0	0	0	0	0	0	-1	0	1	4	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-8

Resta de la matriz de confusión 20vs20 a la matriz de confusión de la Implementación alternativa A para el experimento *exp_lang_3*.

Matrices de confusión *exp_lang_4*

exp_lang_4-ARA	1	2	3	4	5
ara-arz (1)	1664	0	16	3	12
ara-acm (2)	0	708	4	1	24
ara-apc (3)	0	35	756	10	41
ara-ary (4)	0	7	6	590	9
ara-arb (5)	0	0	0	0	61

exp_lang_4-ZHO	6	7	8	9
zho-yue (6)	37	0	5	6
zho-cmn (7)	3	1268	11	23
zho-cdo (8)	3	0	76	2
zho-wuu (9)	2	0	4	68

exp_lang_4-ENG	10	11	12
eng-gbr (10)	7	0	0
eng-usg (11)	8	1655	46
eng-sas (12)	1	0	125

exp_lang_4-FRE	13	14
fre-waf (13)	60	0
fre-hat (14)	0	48

exp_lang_4-QSL	15	16
qsl-pol (15)	469	5
qsl-rus (16)	4	260

exp_lang_4-SPA	17	18	19	20
spa-car (17)	457	1	2	0
spa-eur (18)	1	80	9	0
spa-lac (19)	1	21	34	0
por-brz (20)	0	0	0	8

Matrices de confusión de las evaluaciones por clúster para el experimento *exp_lang_4*.

exp_lang_4 20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1622	0	14	2	7	1	1	3	2	1	0	1	0	21	0	1	5	2	0	12
ara-acm (2)	0	667	3	0	21	7	0	4	1	0	0	0	27	2	0	2	0	3	0	0
ara-apc (3)	0	12	718	4	23	6	0	6	0	2	1	0	45	1	0	2	0	13	9	0
ara-ary (4)	0	3	4	563	7	0	0	6	5	2	0	0	14	3	1	1	0	0	2	1
ara-arb (5)	0	0	0	0	60	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
zho-yue (6)	0	0	0	0	0	37	0	5	6	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	0	0	0	0	0	3	1243	10	22	2	2	1	10	4	0	0	1	0	1	6
zho-cdo (8)	0	0	0	0	0	3	0	72	2	0	0	0	4	0	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	1	0	3	64	0	0	0	5	0	0	0	0	0	1	0
eng-gbr (10)	0	0	0	0	1	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
eng-usg (11)	3	2	0	0	3	2	2	7	11	6	1589	41	1	10	2	2	15	3	4	6
eng-sas (12)	0	0	4	0	0	1	0	5	2	0	0	108	0	0	0	0	0	0	3	3
fre-waf (13)	0	0	0	0	0	1	0	0	0	0	0	0	59	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	47	0	0	0	0	0	1
qsl-pol (15)	0	0	0	0	22	0	0	0	3	4	0	3	18	1	416	0	0	3	4	0
qsl-rus (16)	0	0	2	2	4	0	0	1	1	2	0	0	3	1	3	243	0	1	0	1
spa-car (17)	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	455	1	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	78	9	0
spa-lac (19)	0	0	0	0	0	2	0	0	0	0	0	0	8	0	0	1	1	14	30	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

Matriz de confusión de la evaluación 20vs20 para el experimento *exp_lang_4*.

exp_lang_4 Imp.Alt. A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	1507	0	16	3	7	1	2	3	0	1	7	5	11	61	2	3	51	6	1	8
ara-acm (2)	0	622	3	1	16	0	0	0	0	0	0	1	65	16	1	8	0	3	1	0
ara-apc (3)	0	24	700	7	28	0	0	3	1	0	2	0	49	16	6	4	1	1	0	0
ara-ary (4)	0	3	3	525	7	1	0	0	2	0	0	1	22	16	16	13	0	0	3	0
ara-arb (5)	0	0	0	0	27	0	0	0	0	0	0	0	4	28	0	0	0	0	1	1
zho-yue (6)	0	0	0	0	0	30	0	5	6	0	0	0	6	0	0	0	0	1	0	0
zho-cmn (7)	0	0	0	0	0	3	1255	10	21	0	5	0	7	2	0	0	0	1	1	0
zho-cdo (8)	0	0	0	0	0	0	0	50	1	0	0	0	24	4	1	0	0	1	0	0
zho-wuu (9)	0	0	0	0	0	1	0	3	59	0	0	0	10	0	0	0	0	0	1	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	2	0	0	0	4	1	0	0	0	0	0
eng-usg (11)	0	0	0	0	1	0	4	3	3	7	1629	42	1	7	3	0	8	0	1	0
eng-sas (12)	0	0	1	0	0	1	0	2	0	0	0	77	16	11	2	12	0	0	4	0
fre-waf (13)	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	0	0	0	0	0	0	0	0	2	5	464	3	0	0	0	0
qsl-rus (16)	0	0	1	0	0	0	0	0	0	0	0	0	8	5	4	245	1	0	0	0
spa-car (17)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	456	1	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	78	9	0
spa-lac (19)	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	1	18	31	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0

Matriz de confusión de la evaluación de la implementación alternativa A para el experimento *exp_lang_4*.

exp_lang_4 Imp.Alt. B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	811	0	6	2	7	25	1	29	24	60	8	100	115	181	39	80	45	29	41	92
ara-acm (2)	0	577	3	1	22	20	0	14	5	6	0	2	53	11	1	13	0	5	1	3
ara-apc (3)	0	22	631	4	36	17	0	19	11	8	0	5	44	19	2	4	0	11	7	2
ara-ary (4)	0	3	2	441	9	6	0	7	13	10	0	5	32	16	12	20	0	11	10	15
ara-arb (5)	0	0	0	0	60	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
zho-yue (6)	0	0	0	0	0	37	0	5	6	0	0	0	0	0	0	0	0	0	0	0
zho-cmn (7)	1	0	0	3	11	3	826	9	21	79	1	29	95	94	16	40	15	18	7	37
zho-cdo (8)	0	0	0	0	0	3	0	75	2	1	0	0	0	0	0	0	0	0	0	0
zho-wuu (9)	0	0	0	0	0	2	0	4	66	0	0	0	2	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	0	0	0	0	0	6	0	0	0	1	0	0	0	0	0	0
eng-usg (11)	1	1	0	1	16	63	5	60	41	7	999	29	86	138	40	70	29	24	32	67
eng-sas (12)	0	0	0	0	0	4	0	9	3	0	0	105	1	0	1	2	0	0	0	1
fre-waf (13)	0	0	0	0	0	4	0	0	0	0	0	0	56	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0
qsl-pol (15)	0	0	0	0	6	2	0	0	3	9	0	5	4	2	439	2	0	0	0	2
qsl-rus (16)	0	0	0	0	0	0	0	0	0	3	0	0	7	1	4	248	0	0	0	1
spa-car (17)	0	0	0	0	1	3	1	1	1	0	0	0	2	3	1	4	440	1	2	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	78	9	0
spa-lac (19)	0	0	0	0	0	1	0	3	0	0	0	0	4	0	0	0	1	16	31	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	7

Matriz de confusión de la evaluación de la implementación alternativa B para el experimento *exp_lang_4*.

exp_lang_4 imp.A-20vs20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ara-arz (1)	-115	0	2	1	0	0	1	0	-2	0	7	4	11	40	2	2	46	4	1	-4
ara-acm (2)	0	-45	0	1	-5	-7	0	-4	-1	0	0	1	38	14	1	6	0	0	1	0
ara-apc (3)	0	12	-18	3	5	-6	0	-3	1	-2	1	0	4	15	6	2	1	-12	-9	0
ara-ary (4)	0	0	-1	-38	0	1	0	-6	-3	-2	0	1	8	13	15	12	0	0	1	-1
ara-arb (5)	0	0	0	0	-33	0	0	0	0	-1	0	0	4	28	0	0	0	0	1	1
zho-yue (6)	0	0	0	0	0	-7	0	0	0	0	0	0	6	0	0	0	0	1	0	0
zho-cmn (7)	0	0	0	0	0	0	12	0	-1	-2	3	-1	-3	-2	0	0	-1	1	0	-6
zho-cdo (8)	0	0	0	0	0	-3	0	-22	-1	0	0	0	20	4	1	0	0	1	0	0
zho-wuu (9)	0	0	0	0	0	0	0	0	-5	0	0	0	5	0	0	0	0	0	0	0
eng-gbr (10)	0	0	0	0	-1	0	0	0	0	-4	0	0	0	4	1	0	0	0	0	0
eng-usg (11)	-3	-2	0	0	-2	-2	2	-4	-8	1	40	1	0	-3	1	-2	-7	-3	-3	-6
eng-sas (12)	0	0	-3	0	0	0	0	-3	-2	0	0	-31	16	11	2	12	0	0	1	-3
fre-waf (13)	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
fre-hat (14)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	-1
qsl-pol (15)	0	0	0	0	-22	0	0	0	-3	-4	0	-3	-16	4	48	3	0	-3	-4	0
qsl-rus (16)	0	0	-1	-2	-4	0	0	-1	-1	-2	0	0	5	4	1	2	1	-1	0	-1
spa-car (17)	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0
spa-eur (18)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
spa-lac (19)	0	0	0	0	0	-2	0	0	0	0	0	0	-2	0	0	-1	0	4	1	0
por-brz (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-8

Resta de la matriz de confusión 20vs20 a la matriz de confusión de la Implementación alternativa A para el experimento *exp_lang_4*.