**MEMORIA AMPLIACIÓN PRÁCTICA 4 – RECONOCIMIENTO Y VERIFICACIÓN DE UN LOCUTOR**

1. INTRODUCCIÓN

En la práctica 4, se realizó un sistema de reconocimiento y verificación de locutor, mediante el uso de clasificadores GMM, modelando cada locutor como una combinación de gaussianas. Para ajustar los parámetros de las gaussianas, se utiliza el método de *“Expectation Maximization”* (EM), en que se alternan procesos de esperanzar y de maximización para encontrar estimadores de máxima verosimilitud. Para la inicialización de los GMMs, tenemos diferentes métodos a escoger: método VQ (cuantización vectorial), método EM, o aleatoriamente.

Para poder clasificar correctamente, disponemos de una base de datos con muestras de diferentes hablantes, que entrenaremos para poder clasificar y verificar.

En la práctica 4, se probó el sistema para distintos parámetros: LP, LPC y MFCC. En esta ampliación, vamos a calcular las características dinámicas, la primera y segunda derivada de los coeficientes MFCC, y vamos a probar el sistema para cada una de estas características.

1. DESCRIPCIÓN E IMPLEMENTACIÓN DEL DISEÑO

Para ejecutar los programas que se irán explicando a continuación utilizamos el script ‘run\_skpid\_din.sh’ que contiene las llamadas a las funciones correspondientes.

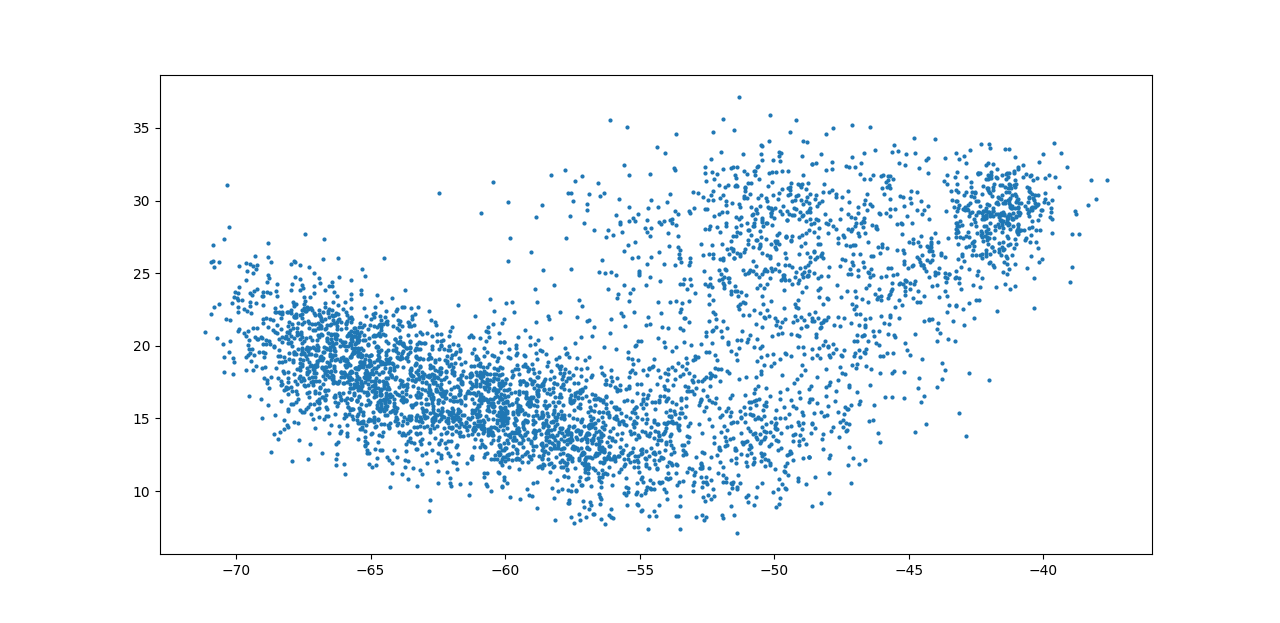
Para empezar, necesitamos extraer las características de cada trama de voz de los datos de la base de datos. En este caso, vamos a calcular los coeficientes delta (primera derivada de MFCC) y delta-delta (segunda derivada de MFCC), para hacer una comparación con los resultados obtenidos anteriormente en la práctica 4, para las características LP, LPC y MFCC.

Para el cálculo de los coeficientes dinámicos, definimos los ficheros ‘wav2mfcc\_delta’ y ‘wav2mfcc\_deltadelta’, dónde hacemos una extracción de las características MFCC y calcula sus derivadas, utilizando la librería SPTK.

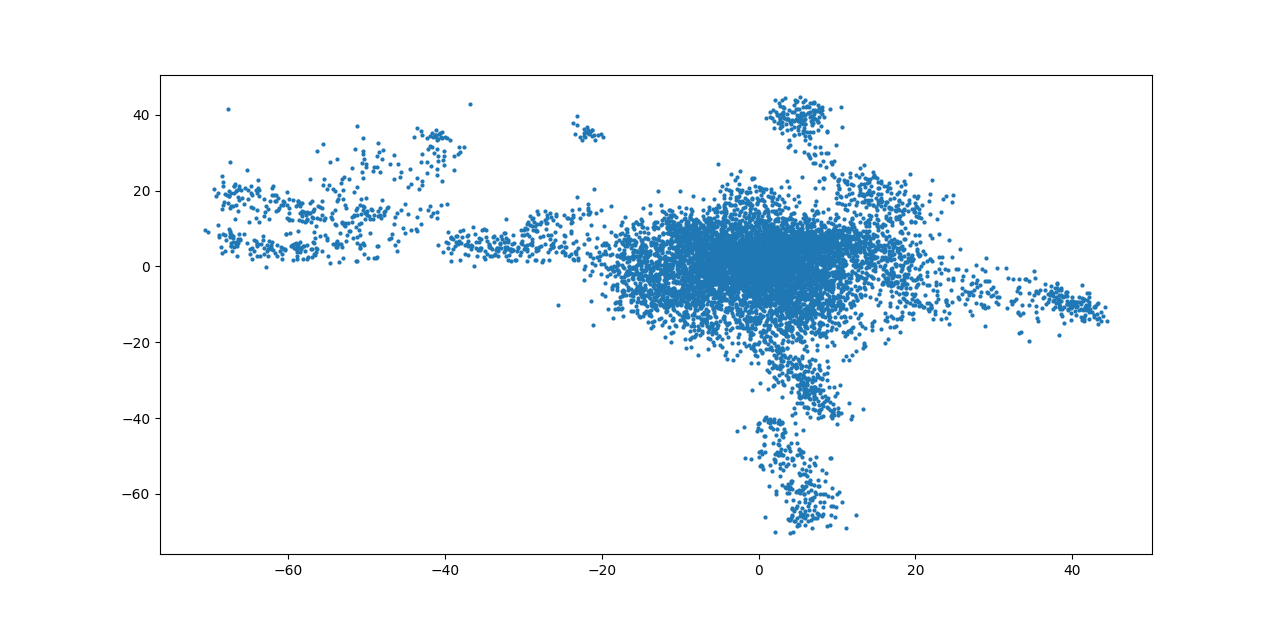
En el fichero ‘run\_skipd\_din.sh’, definimos dos funciones *‘compute\_mfccdelta()*’ y *‘compute\_mfccdeltadelta()’*  para extraer las características.

A continuación, se muestra la representación de la dependencia entre los coeficientes 2 y 3 de las tres parametrizaciones para una señal de prueba.

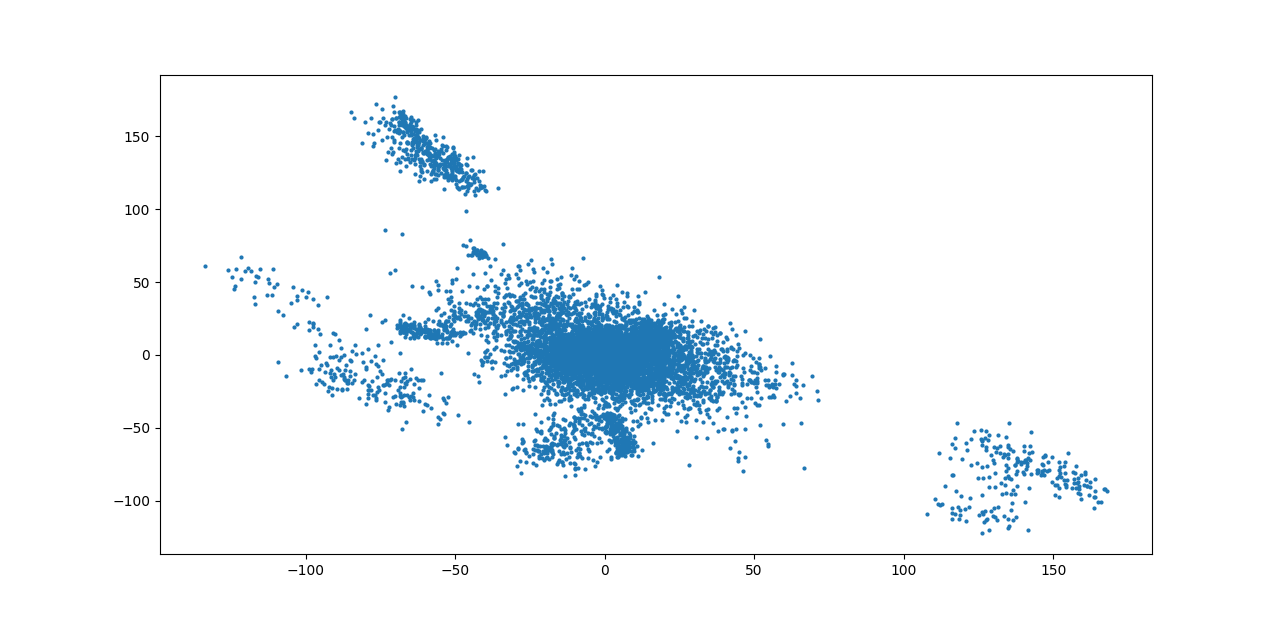
Si nos fijamos, en las parametrizaciones delta y delta-delta los coeficientes están más correlados, por lo que a partir de un coeficiente podemos determinar otros, por lo que hay menos información. Podemos intuir que los resultados que obtendremos para estos coeficientes serán peores que para otras características, ya que tendremos menos información en los casos de las derivadas.



.Parametrización MFCC



. Parametrización Delta (primera derivada de MFCC)

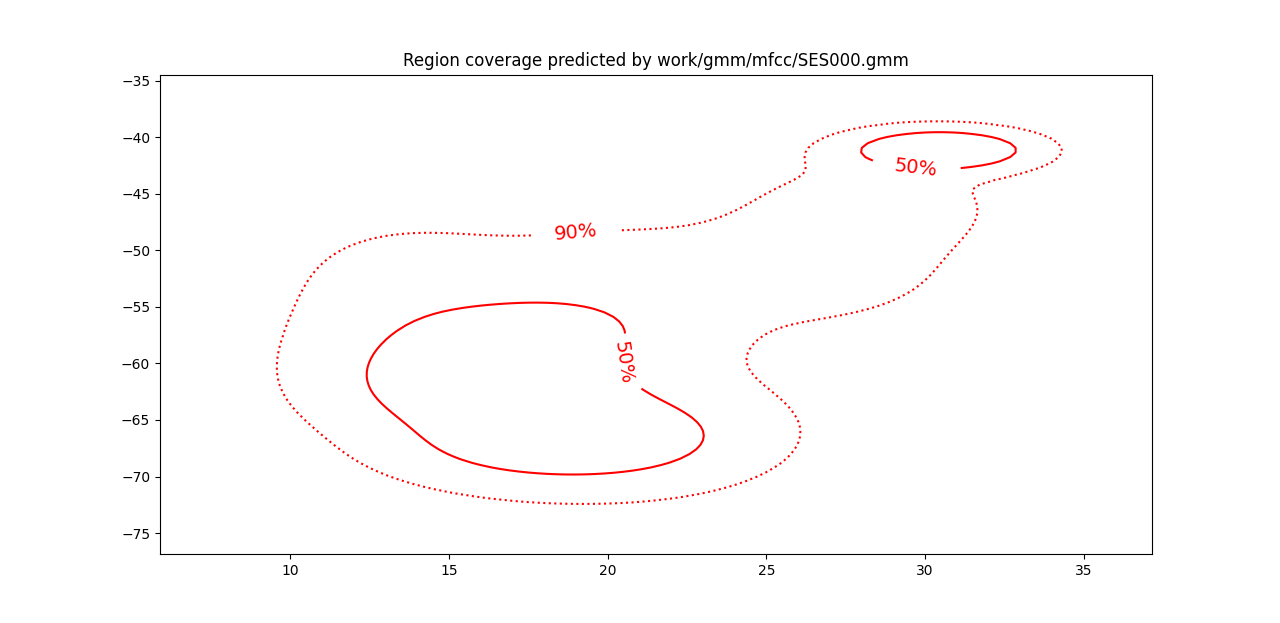


. Parametrización delta-delta (segunda derivada de MFCC)

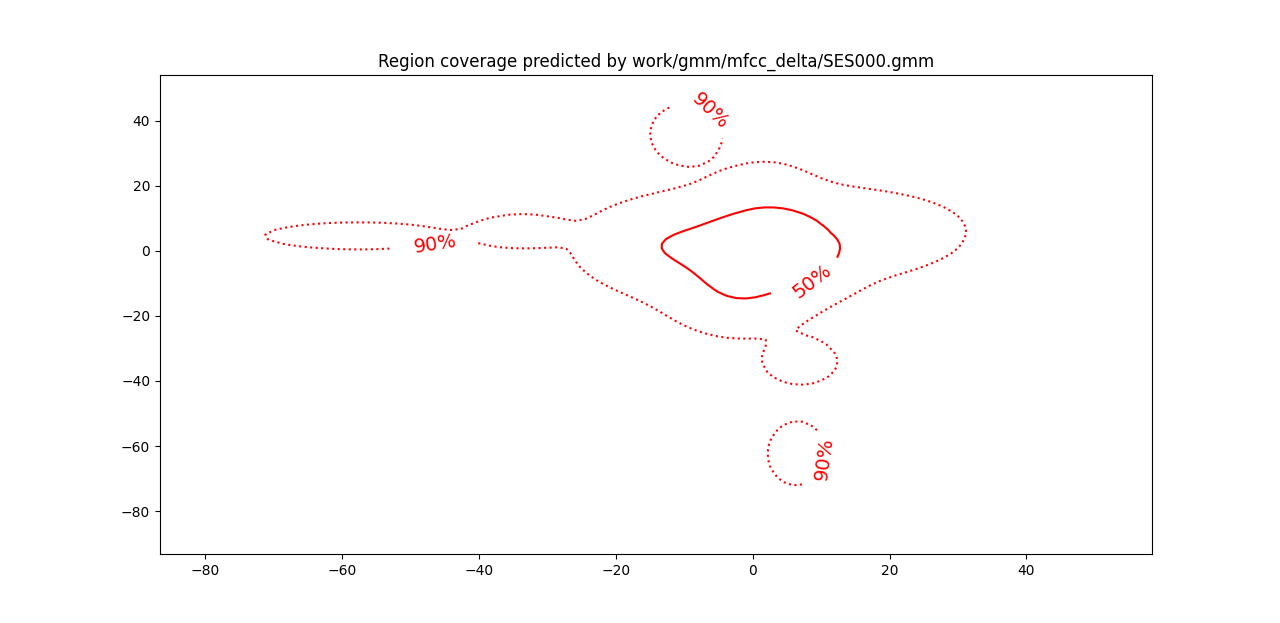
Una vez hemos extraído las características de los locutores, necesitamos generar los modelos GMM. La inicialización de los GMMs puede ser aleatoria, EM o VQ. Los mejores resultados los ha dado la inicialización VQ, por lo que va a ser la inicialización usada. Para generar los GMM, llamamos a la función ‘*train’* de ‘run\_skipd\_din.sh’, donde hacemos un entramiento para cada hablante.

También llamamos a la función ‘*trainworld’* que necesitamos para hacer la verificación al final. En este caso, entrenamos los GMM con las características de un conjunto de hablantes más amplio, que corresponden a un modelo del “mundo”, donde incluye usuarios legítimos e impostores.

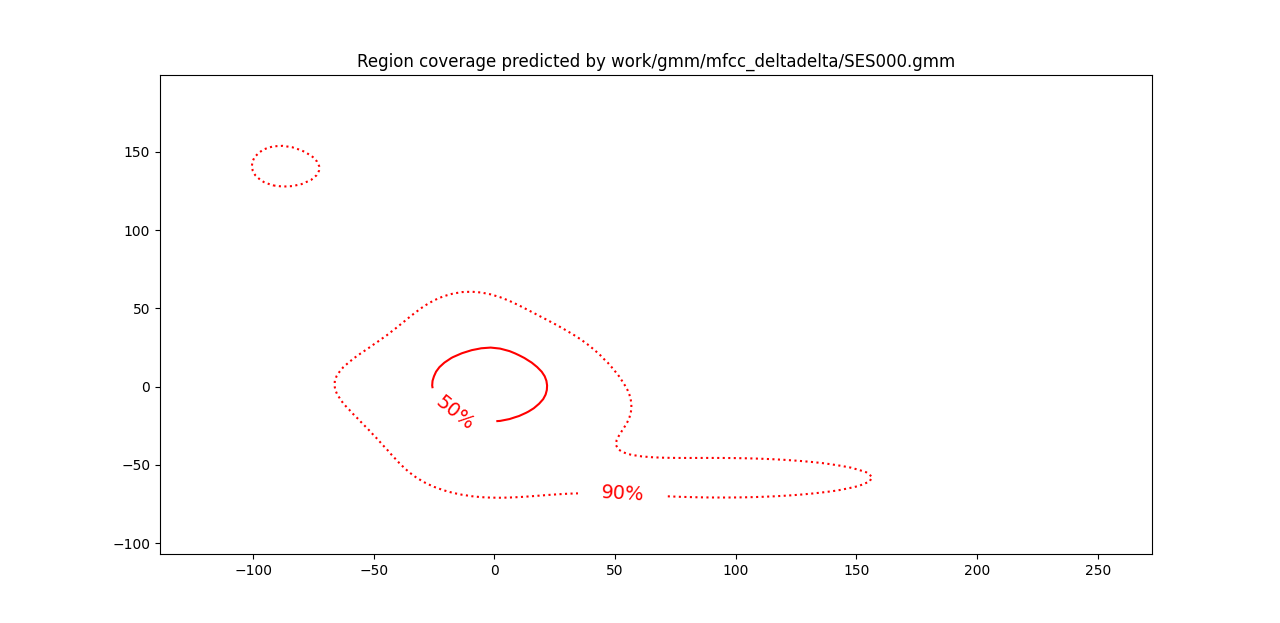
A continuación, se muestra la representación de las funciones de densidad de probabilidad modeladas por los GMMs de un mismo locutor para los dos primeros coeficientes de MFCC, delta i delta-delta. Aquí también podemos observar que para el mismo locutor, para las deltas, la clasificación será peor, ya que las regiones varían de los coeficientes MFCC, los delta y los delta-delta.



. Función de densidad de probabilidad modelada por los GMMs de MFCC



. Función de densidad de probabilidad modelada por los GMMs de ΔMFCC



.Función de densidad de probabilidad modelada por los GMMs de Δ2MFCC.

Una vez hecho el entrenamiento, llamamos a la función ‘gmm\_*classify’*  a través de ‘*run\_skipd test’* para hacer la clasificación del locutor. Esta función calcula la probabilidad que cada muestra corresponda a una gaussiana y se compara con la clase a la que realmente pertenece.

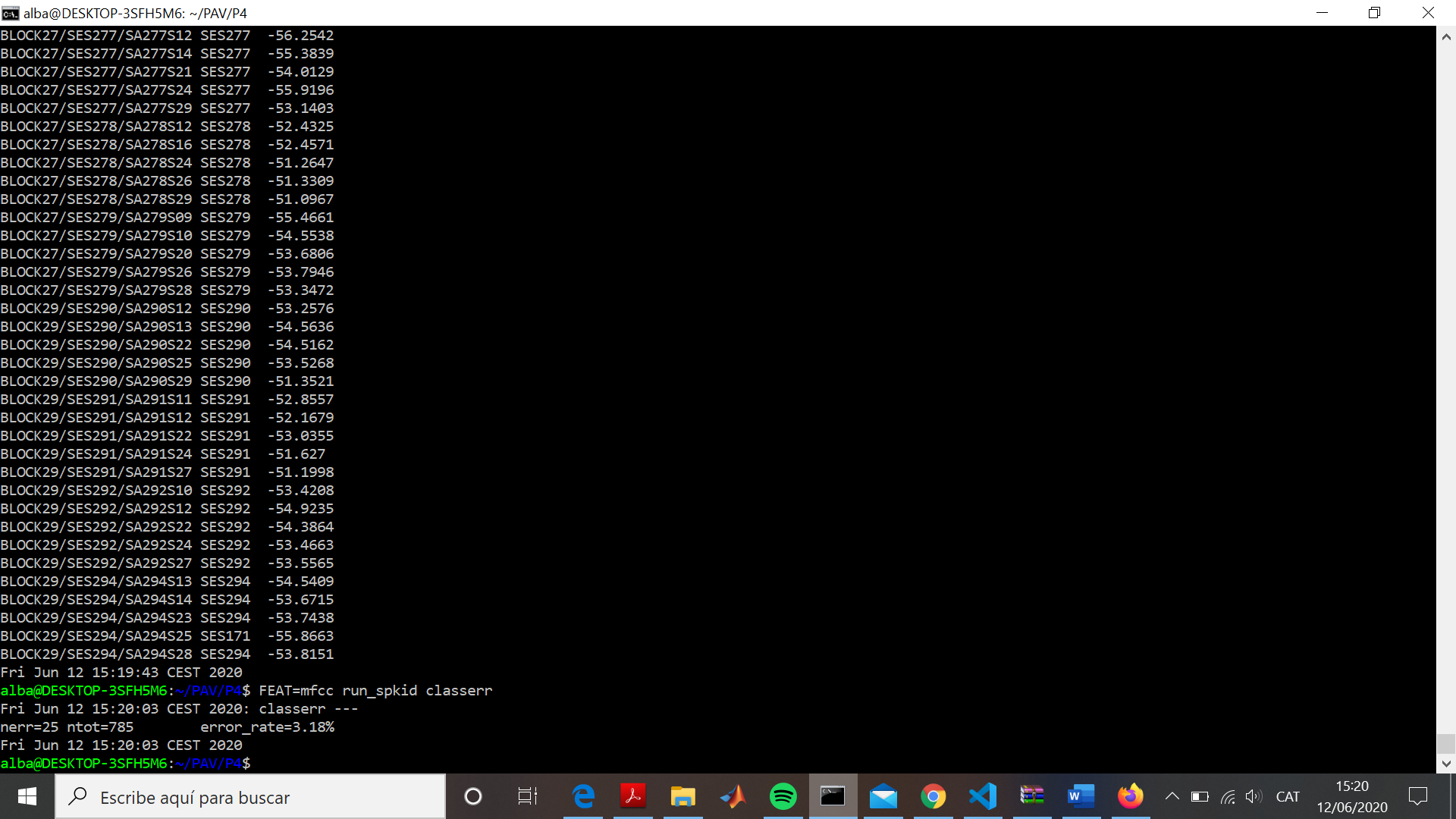
Y después hacemos la verificación, llamando a la función ‘verify’ del ‘run\_skipd.sh’, que calcula las probabilidades del usuario y del mundo y las compara para calcular el score. Este score servirá para aceptar, es decir, que sea un usuario legítimo o rechazar al locutor según un umbral. Esta verificación tiene dos errores: Missed y FalseAlarm. Missed es cuando se le niega el permiso a un usuario legítimo y FalseAlarm es cuando se acepta un impostor.

1. EVALUACIÓN DE LOS RESULTADOS

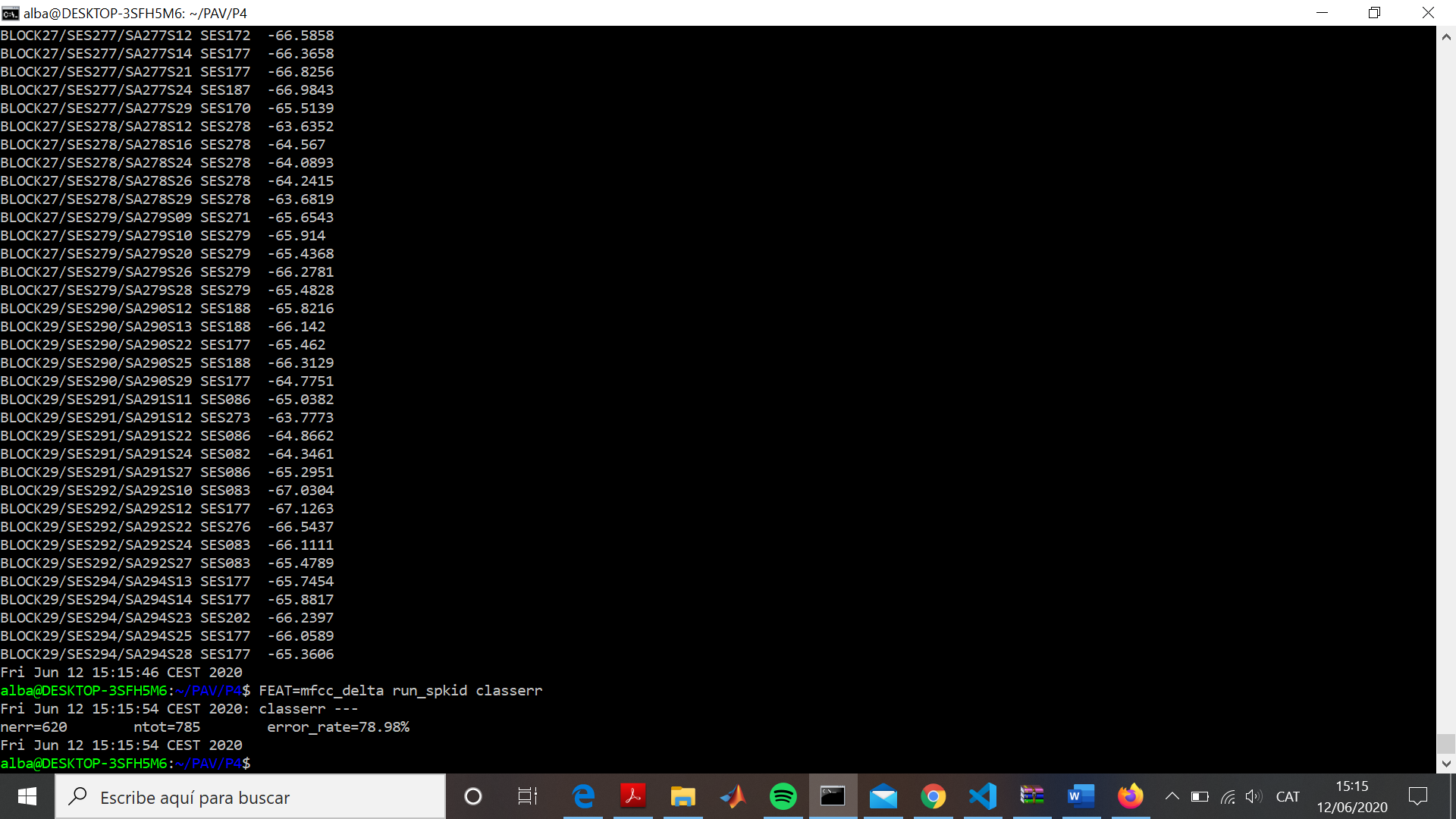
Para evaluar la clasificación y verificación realizadas anteriormente, hemos definido dos funciones en el fichero ‘*run\_spkid\_din.sh’*: *classerr y verif\_err*.

La función ‘*classerr’* compara los resultados de la clasificación hecha por nuestro sistema con los datos de los usuarios para ver el número de errores.

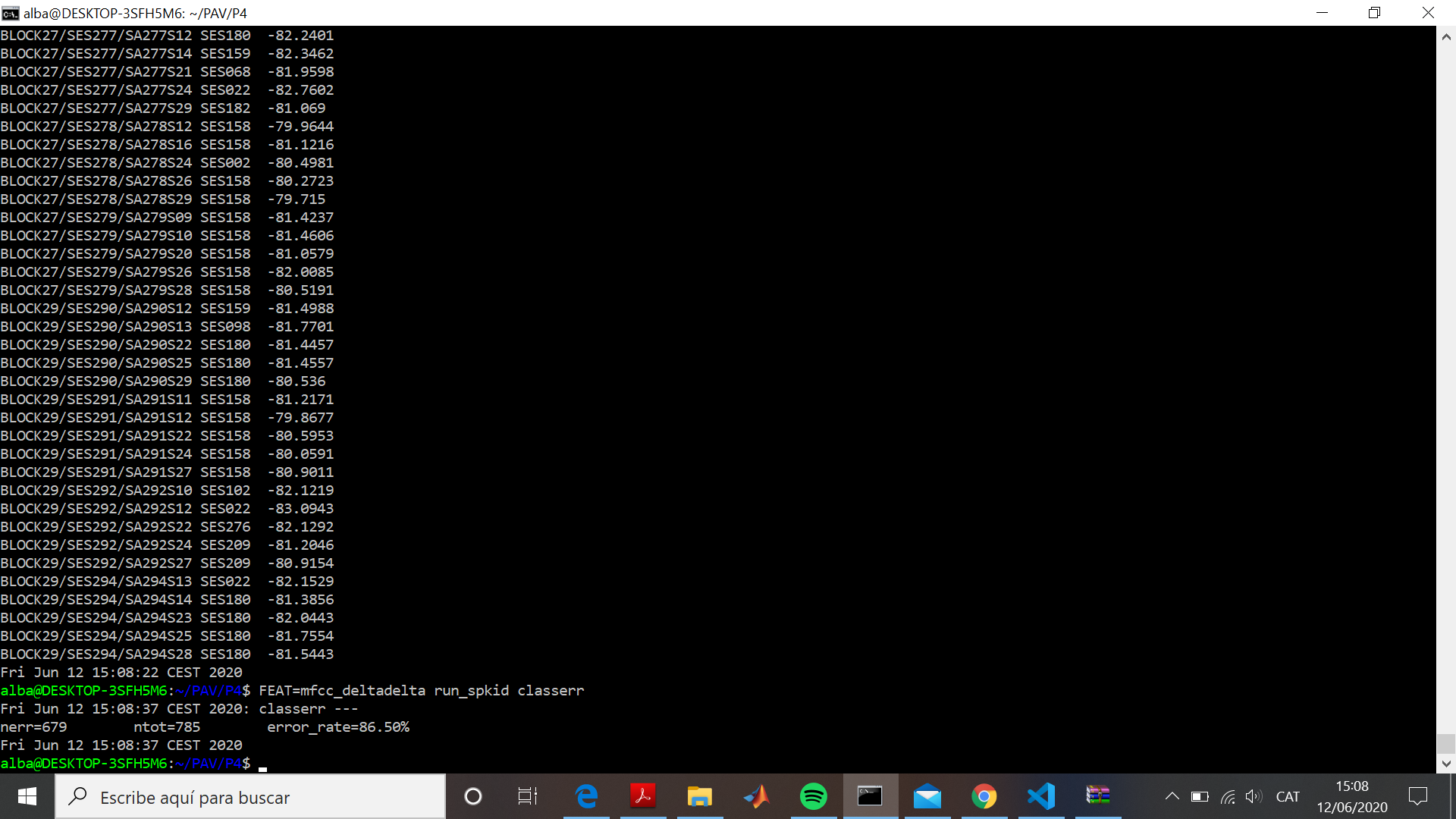
Para la aplicación de los coeficientes dinámicos y los MFCC hemos obtenido los siguientes resultados:



. Errores en la clasificación para los MFCC



. Errores en la clasificación para los ΔMFCC



. Errores en la clasificación para los Δ2MFCC

La función *‘verif\_err’* evalua la calidad del sistema y calcula los *Missed*, los *FalseAlarm* i el valor del umbral *(THR*).

Para la aplicación con los coeficientes dinámicos y los MFCC hemos obtenido los siguientes resultados:

==============================================

THR: 2.33881544478942

Missed: 145/250=0.5800

FalseAlarm: 0/1000=0.0000

----------------------------------------------

==> CostDetection: 58.0

==============================================

. Resultados de la verificación con MFCC

==============================================

THR: 2.87442088543181

Missed: 242/250=0.9680

FalseAlarm: 0/1000=0.0000

----------------------------------------------

==> CostDetection: 96.8

==============================================

. Resultados de la verificación con ΔMFCC

==============================================

THR: 2.1523061820168

Missed: 247/250=0.9880

FalseAlarm: 0/1000=0.0000

----------------------------------------------

==> CostDetection: 98.8

==============================================

12. Resultados de la verificación con Δ2MFCC

1. DISCUSIÓN DE LOS RESULTADOS

Los resultados de nuestra aplicación usando los coeficientes MFCC son aceptables, con una tasa de error del 3.18%. Si sustituimos los MFCC por sus derivadas, por la primera y por la segunda por separado, la aplicación empeora muchísimo. No podemos usar los coeficientes dinámicos por si solos ya que no implican una mejora de la clasificación y reconocimiento del locutor. Para el uso de los coeficientes delta, la tasa de error es de casi un 80% y para los delta-delta, un error del 86.5%. El error de clasificación es muy grande, por lo que no podemos sustituir los coeficientes MFCC, u otros como el LP o LPCC, por las características dinámicas.

Respecto a la verificación, el coste para los MFCC es alto, mas de un 58, pero para las características dinámicas, un 96.8 y 98.8, para las primeras y segundas derivadas respectivamente.

Des de un inicio, al representar la dependencia entre coeficientes de las características dinámicas, ya se podía prever que el uso de las características dinámicas por sí solas, no iban a mejorar el reconocimiento y clasificación del locutor, ya que contienen menos información.

1. CONCLUSIONES

En definitiva, sustituir los coeficientes MFCC, LP o LPCC, por las características dinámicas, no es una solución de mejora para nuestro sistema, ya que la clasificación y reconocimiento de locutores empeora muchísimo. Respecto a la inicialización de los GMMs, el mejor resultado lo obtenemos para la inicialización VQ (cuantificación vectorial).

Para futuras aplicaciones, otra implementación, podría ser la implementación de los coeficientes MFCC y sus características dinámicas juntos, y comprobar si su aplicación conjunta supone una mejora de la aplicación.