Práctica 4 - Clasificación y verificación del hablante

1st Jaume Colom Hernandez
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
jaumecolomhernandez@gmail.com

2nd Miquel Martinez Blanco
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
mbmiquel@gmail.com

***Abstract*—En esta práctica se implementará un clasificador de locutores y una aplicación para la verificación del hablante mediante programación en C++ y bash.**

***Index Terms*—Procesado, audio, voz, C++, bash, programación.**

I. Introducción

El objetivo de esta práctica es implementar un sistema de clasificación y verificación de hablante. Hemos programado todas las partes de este sistema: extracción de características de la voz, modelado probabilístico con GMMs, clasificador MAP, verificador con un threshold y la verificación de los resultados en los dos programas. Todos estos pasos se han automatizado mediante scripts de Bash.

Para la parte opcional hemos realizado un sistema de clusterización automática de hablantes. Tambien como ampliación portamos todo el codigo en C++ a Python.

II. Extracción de características

El primer paso en un sistema de procesado de voz es la extracción de características. Consiste en procesar los audios para obtener información con la que posteriormente entrenar los sistemas.

Para nuestro sistema hemos hecho pruebas con tres tipos de características del audio: LPC, LPCC y MFCC. A continuación vamos a explicar cada una de ellas y discutir cual es la mejor para nuestro sistema.

A. Coeficientes LPC

La idea básica de la predicion lineal es que la muestra actual de audio se puede aproximar como una combinacion lineal de las anteriores muestras.
Expresion directa del modelo fuente-filtro:

$$s[n] = \sum_{k=1}^{p} = a_k s[n-k] + e[n]$$

Tenemos que minimizar el error residual e[n]:

$$\sum_n e[n]^2 = \sum_n (s[n] - \sum_{k=1}^{p} = a_k s[n-k])^2$$

Si derivamos a_k y solucionamos llegamos a:

$$r_{ss}(|j-k|)[n] = \sum_{n}^{p} = a_k s[n-k]$$

En el siguiente gráfico mostramos el plot del coeficiente 2 y 3 de un audio. Cada punto es una trama de 15ms de la que hemos calculado 9 coeficientes LPC. Observamos que los puntos están altamente correlados.

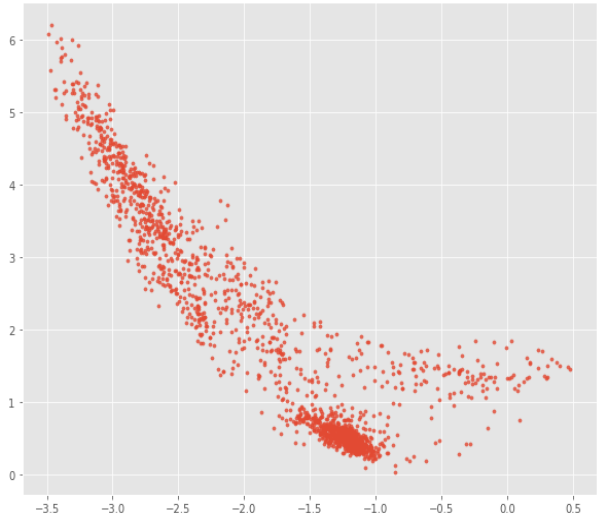

Fig. 1. Autocorrelación de los coeficientes LPC

B. Coeficientes LPCC

El análisis mediante LPCC se basa en asumir que la forma del tracto vocal modela el sonido producido. La manera de obtener estos coeficientes es a partir de las siguientes transformaciones

$$c_0 = ln(p)$$
$$c_1 = a_1$$
$$c_i = -a_i + \sum_{n=1}^{i-1} \frac{n a_{i-n} c_n}{i}$$

Fig. 2. Fórmulas para la obtención de los coeficientes LPCC

Tal y como se puede apreciar en la imagen inferior estos nuevos coeficientes están más dispersos que los LPC por lo tanto son menos correlados, eso quiere decir que nos están aportando más información y dan unos mejores resultados a la hora de efectuar la clasificación de hablantes.

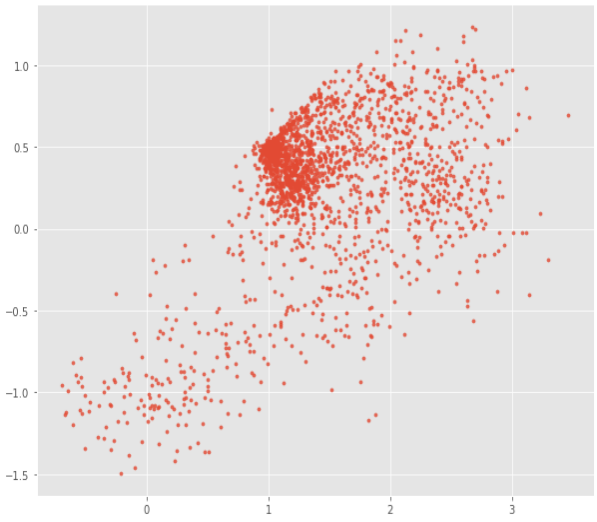

Fig. 3. Autocorrelación de los coeficientes LPCC

C. Coeficientes MFCC

Los coeficientes MFCC son los mas utilizados para reconocimiento del habla ya que estos tienen en cuenta el conjunto de frecuencias en las cuales el oído humano es más sensible. Para obtener estos coeficientes se hace pasar la energía de la señal de voz a través de un banco de filtros de Mel y a continuación se efectúa la transformada discreta de coseno (DCT) y se guardan los coeficientes los cuales recibirán el nombre de MFCC. Como se puede observar en la figura inferior la correlación entre coeficientes es mucho menor y hay una mejor dispersión, esto aporta mucha más información y es por ello que son los coeficientes que dan unos mejores resultados.

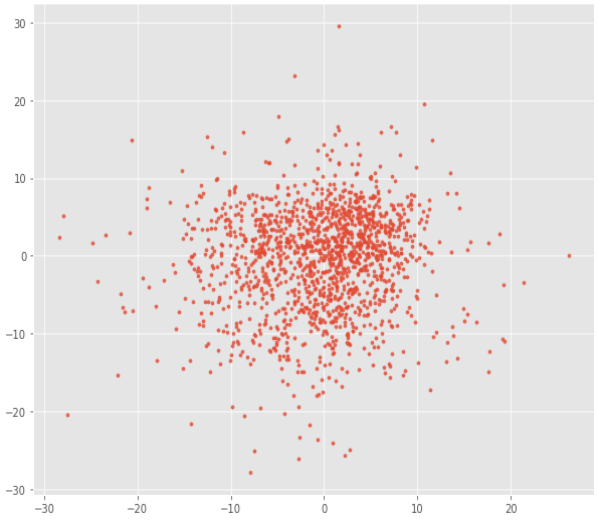

Fig. 4. Autocorrelación de los coeficientes MFCC

D. Discusión extracción de características

Los datos que extraemos mediante los coeficientes LPC, LPCC y MFCC se usan para el entrenamiento de los modelos de clasificación y verificación. El objetivo es que estos modelos sean capaces de reproducir con la máxima exactitud las voces con las que trabajamos, por lo tanto los datos que usemos tienen que contener el máximo de información. Viendo los plots de los coeficientes 2 y 3 del mismo audio de los diversos coeficientes se observa que los LPC tienen un correlación alta entre ellos, los LPCC también están correlados pero bastante menos ya, los coeficientes MFCC no tienen prácticamente correlación. Hemos usado dos coeficientes contiguos (N y N+1) porqué al estar más juntos són más correlados por definición, para este análisis nos interesa utilizar el peor caso.

III. Modelado probabilístico con GMM

A. Explicación GMMs

El modelado probabilístico con GMM consiste en una función paramétrica de densidad de probabilidad, que esta compuesta por una suma de Gaussianas ponderadas. Para cada hablante se modela su GMM a partir de unos coeficientes MFCC, extraídos de los audios de entrenamiento. Por lo tanto para cada audio, a partir de los coeficientes, la función de GMM nos retornará la probabilidad de que ese audio pertenezca a cada uno de los locutores. Formula de la Gaussiana multivariable:

$$p(x) = \sum_{k=1}^{K} c_k p(a_k|m_k) \tag{1}$$

Fig. 5. Modelado probabilístico de Gaussianas

c_k representa los pesos de las Gaussiana y $p(a|m_k)$ la probabilidad de que el conjunto de coeficientes `a` pertenezcan a la Gaussiana m_k.

B. Explicación algoritmo EM

El algoritmo de expectation-maximization (EM) es el que nos permite entrenar el modelo GMM de cada locutor, este consiste en estimar los pesos, medias y varianzas de las Gaussianas a partir de los coeficientes que introducimos en la entrada.
El funcionamiento consiste en que en cada iteración se calculan los pesos, medias y varianzas a partir de la probabilidad a priori de un conjunto de coeficientes de entrenamiento. Con estos nuevos parámetros se modifica el modelo GMM que utilizará el algoritmo en la siguiente iteración. Este proceso se realiza hasta que el sistema converge.

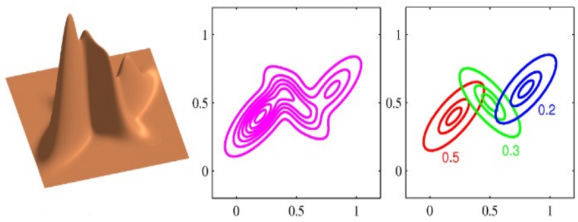

Fig. 6. Modelo de GMM de tres Gaussianas de un locutor

IV. Clasificación de locutor

El objetivo del clasificador es determinar a que locutor pertenece cada uno de los audios que nosotros le pasamos, utilizando para ello el modelado de GMMs.

A. Esquema funcionamiento

1 Abrimos la base de datos de audios.
2 Se crea una lista con todos los nombres de los archivos de audio que hay dentro.
3 Se divide esa lista en dos, siendo una la que contiene los audios que entrenaran el modelo de GMM y los otros los de test para la evaluación del sistema.
4 A continuación se van calculando los coeficientes LPCC o MFCC de cada uno de los audios de la lista de entrenamiento.
5 Se entrena el modelo de GMMs de cada locutor mediante el algoritmo de EM.
6 A continuación se ejecuta el programa de clasificación proporcionado, el cual para cada audio de la lista de test nos devuelve el locutor más probable.
7 Se muestran por pantalla, el número total de audios, la cantidad que han sido mal clasificados y el tanto por ciento de error.

B. Decisor MAP

El clasificador implementado hace uso del criterio MAP, máximo a posteriori, ya que calcula la probabilidad de cada uno de los locutores a partir de los coeficientes de entrada con un conocimiento previo de la distribución de esos coeficientes según el locutor.

V. Verificación de locutor

El objetivo de este segundo programa es comprobar a partir de un audio si la voz es de un supuesto locutor. Esto se podría aplicar por ejemplo a banca telefónica para dar o no acceso a un locutor a los datos de la que dice que es su cuenta bancaria. Debido al tipo de sistema se debe considerar que es mucho peor dar acceso a un usuario impostor (falsa alarma) que dar un fallo de detección (miss) y denegar a un usuario legitimo su acceso.

A. Esquema funcionamiento

1 Abrimos la base de datos de audios.
2 Se crean 2 listas de audios: entrenamiento y test.
3 Usando los audios de entrenamiento se genera un modelo GMM de cada usuario.
4 Se ejecuta el programa de validación el cual asigna a cada audio un posible locutor aleatorio de la lista de test (este puede ser el usuario legitimo o un impostor), a continuación el programa tiene que indicar si ese usuario aleatorio corresponde con el correcto.
5 Ampliación: Para una mejora en la precisión del calculo de las probabilidades todo se divide entre la probabilidad del `World`. Por lo tanto también se debe generar el modelo GMM del mundo.

B. Variante con el World

Debido a que el `score` es un valor muy variante lo que se suele hacer es normalizar con la probabilidad de la señal respecto del mundo. Esto consiste en caracterizar el conjunto de locutores mediante una sola GMM, esto ayuda a mejorar mucho la precisión del sistema.

C. Verificador threshold

El sistema coge los coeficientes de ese audio y los compara con la GMM de entrenamiento de ese usuario, de manera que si la similitud supera un cierto `threshold` se da por valido el acceso a ese usuario.

D. Función de coste

Se ha definido como falsa alarma dejar acceder al sistema a un impostor y como fallo del sistema no permitir el acceso a un usuario legitimo.
Puesto que no tiene la misma gravedad un fallo del sistema que una falsa alarma se ha creado una función de costes, en la cual se le ha dado mucha mas importancia a las falsas alarmas con un mayor coste. A la hora de evaluar los posibles sistema debemos escoger aquel cuyo coste obtenido sea menor.

$$C = \frac{1}{min(p,1-p)}(pp_m + (1-p)p_{_{fa}})100 \quad (2)$$

Fig. 7. Función de coste

VI. Ampliación: Clustering de locutor

El problema de clustering de locutor consiste en agrupar audios no etiquetados de hablantes diferentes en grupos de un mismo hablante. La principal dificultad es que es totalmente no supervisado, es decir no se entrena el modelo con datos a priori de los hablantes, solo se usan los mismos datos a clusterizar.

A. Esquema funcionamiento

El esquema de nuestro sistema de clustering es el siguiente:

1 - Lectura y extracción de características de los datos
2 - Modelado de cada audio con GMM
3 - Calculamos la logprob de cada audio con cada audio
4 - Juntamos los dos audios que tienen la score más alta (con matices).
5 - Volver al paso 2 hasta que quedén 2 clusters.

B. Visualización

En el siguiente gráfico vemos visualizados la logscore de cada audio con cada audio. En el eje vertical tenemos las GMM de cada audio y en el horizontal todos los audios, de forma que cada cuadradito es una representación visual de la logscore. Para este caso se usan 40 audios, 20 del hablante 1 y 20 del hablante 2, los audios están ordenados por hablante.

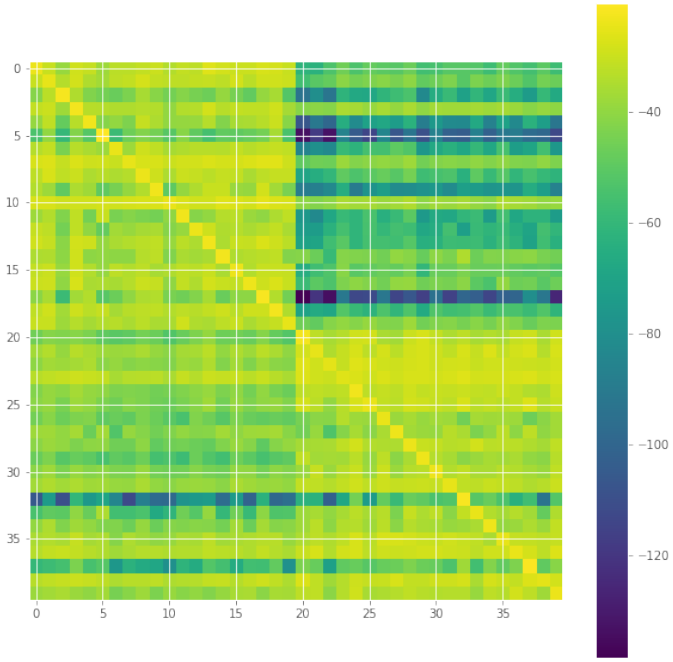

Fig. 8. Visualización clustering

Se aprecia una partición del gráfico en 4 cuadrantes, esto se debe a que los audios están ordenados. Tiene sentido que los 20 primeros audios tengan una puntuación más alta (color más claro) con las 20 primeras GMM ya que estas son las que corresponden a sus audios. Apreciamos que la linea diagonal es muy clara, esto porque al calcular la logprob del mismo audio con la que entrenamos la GMM pues sale una puntuación muy alta.

C. Iteración

Para clusterizar usamos un clustering bottom-up, en el que se parte de todos los audios individuales y se van juntando uno a uno. El criterio para juntar consiste en juntar los audios con mayor logprob. Pero hay dos condiciones que se tienen que cumplir:

1 - No se puede juntar el mismo audio/grupo de audios con el mismo
2 - No se puede juntar el grupo más grande con otro. A partir de que el grupo más grande tenga 5 audios.

La segunda norma no es obvia así que voy a explicar porqué la definimos. Haciendo experimentos nos encontramos con que el programa terminaba con un cluster muy grande y uno muy pequeño, aveces el cluster pequeño consistía de un solo audios y el grande de todos los otros. Al aplicar esta norma a la hora de juntar los clusters lo que evitamos es que los clusters grandes cojan protagonismo y se lleven todos los audios. Esta norma causa que se generen varios grupos de clusters reducidos, que hacía las últimas iteraciones no se empiezan a juntar. El problema se causa por la poca información a la hora de entrenar las GMMS. Como entrenamos un audio por cada GMM, las GMM tienen poca información aunque suficiente como para discriminar si un audio es del mismo hablante o no. Cuando se genera un cluster grande, al reentrenar las GMM, tenemos una GMM que modelan mejor un hablante y se llega al punto que una GMM entrenada con un grupo de varios audios del mismo locutor da una logscore más alta que una GMM entrenada con un solo audio del locutor correcto.

D. Finalización

Finalizamos el proceso de clusterización cuando se converja a dos clusters. Si realizáramos otra iteración nos quedaríamos con un cluster con todos los hablantes. Este modelo no contemple el caso que tengamos distinto numero de audios por locutor o un numero impar de audios de cada hablante.

VII. Ampliación: Implementación Python

Para entender completamente el sistema de clasificación y verificación de locutor decidimos reimplementar todo el sistema en Python. Aparte de entender mejor el sistema hemos practicado mucho la programación en Python y el uso sus librerias.

A. Librerías usadas

Python tiene un ecosistema de librerias orientadas al uso cientifico muy buenas y faciles de usar. Hemos usado varías de ellas para implementar el sistema.

1 Jupyter Notebook- Esto no es una librería en si. Es una aplicación web que permite ejecutar codigo en tiempo real y combinarlo con un formato de celdas muy cómodo, también se puede introducir MarkDown para explicar el contenido.
2 Librosa- Esta es una librería desarrollada por el laboratorio de reconocimiento y organizacion del locutor de la universidad de Columbia (labROSA). La hemos usado para la lectura y extracción de características MFCC.
3 Numpy- Numpy es el core de todos los paquetes científicos de Python, proporciona arrays multidimensionales de alto rendimiento y herramientas de procesado. Se parece mucho al Matlab.
4 Sklearn- Es una librería de machine learning para python, tiene buenas implementaciones de muchos algoritmos.La hemos usado para implementar el modelado GMM.

VIII. Evaluación del sistema

Tanto en clasificación como en verificación es muy importante usar una lista de audios totalmente distinta a la de entrenamiento para poder evaluar el sistema correctamente. Esto es debido a que con los tests de evaluación queremos medir la capacidad de generalizar de nuestros sistemas con información nunca antes vista.

A. Clasificación de locutor

Tal y como se ha desarrollado en apartados anteriores en el caso de clasificación dada un lista de audios de test, se calcula con cada audio la probabilidad de pertenecer a cada uno de los posibles locutores y se decide el hablante que ha obtenido una mayor probabilidad.
Por pantalla se muestra el numero total de clasificaciones efectuadas y cuantas han sido erróneas. Por lo tanto será mejor sistema aquel que obtenga un menor número de errores.

B. Verificación de locutor

En verificación se intenta acceder con usuarios legítimos o impostores y se contabiliza la cantidad de falsas alarmas o fallos de detección que se han efectuado. Puesto que no tiene el mismo nivel de gravedad dejar acceder a un impostor, que no permitir a un usuario legitimo acceder se ha utilizado una función de coste que permita comparar distintos sistemas y así escoger aquel que tiene un menor coste.

C. Clustering de locutor

Para clustering calculamos las métricas de precision y recall. Esto consiste en contar de los audios escogidos cuales corresponden al grupo correcto y de todos los disponibles cuales recogemos en el mismo grupo. Para calcular las métricas del modelo realizamos diez iteraciones con personas distintas.

IX. Discusión de los resultados

A continuación detallaremos los resultados obtenidos con nuestro programa, analizando cualitativamente y cuantitativamente las diferentes mejoras/pruebas realizadas hasta llegar al modelo final con el mejor rendimiento.

A. Modelo clasificación con LPCC

Tras diversas pruebas se ha encontrado que la manera óptima de obtener el mínimo error es configurando nuestro sistema para que la extracción de coeficientes de LPCC sea de orden 15 y con un total de 17 cepstrums.

```
vie may 17 17:28:40 CEST 2019: classerr ---
nerr=4  ntot=778        error_rate=0.51%
```

Fig. 9. Resultados clasificación con LPCC

Con un error del 0.51% podemos afirmar que la clasificación que se lleva a cabo es muy buena.

B. Modelo clasificación con MFCC

En el caso de los coeficientes MFCC se ha obtenido unos resultados óptimos con un sistema de orden 16. En este caso la diferencia con respecto a los LPCC no ha sido muy notoria.

```
vie may 17 19:10:21 CEST 2019: classerr ---
nerr=3  ntot=784        error_rate=0.38%
```

Fig. 10. Resultado clasificación con MFCC

C. Modelo verificación

Para el programa de verificación se han usado tanto los LPCC como los MFCC siendo estos últimos los que han dado un mejor resultado. Tal y como se puede observar en las imágenes inferiores, los costes son muy elevados debido a la gran cantidad de fallos de detección. Esto es un indicativo de que el treshold es muy restrictivo y pocos usuarios son capaces de ser aceptados por el sistema.

```
============================================
THR: 38.20686813967
Missed:     231/250=0.9240
FalseAlarm: 0/1000=0.0000
--------------------------------------------
==> CostDetection: 92.4
============================================
============================================
THR: -21.23307347767
Missed:     195/250=0.7800
FalseAlarm: 0/1000=0.0000
--------------------------------------------
==> CostDetection: 78.0
============================================
```

Fig. 11. Resultados verificación con LPCC y MFCC respectivamente

D. Modelo verificación con mundo

Con el objetivo de mejorar los resultados previos se genera una GMM del mundo que contiene a todos los locutores y que ayudará a normalizar las probabilidades. En este caso se ha decidido generar la GMM del mundo con 12 Gaussianas. Tal y como se puede observar en las imágenes inferiores añadir esto nos proporciona unos resultados mucho mejores.

```
============================================
THR: 1.21494292523782
Missed:     16/250=0.0640
FalseAlarm: 1/1000=0.0010
--------------------------------------------
==> CostDetection: 16.3
============================================
============================================
THR: 0.254783679977603
Missed:     6/250=0.0240
FalseAlarm: 1/1000=0.0010
--------------------------------------------
==> CostDetection: 12.3
============================================
```

Fig. 12. Resultados verificación mundo con LPCC y MFCC respectivamente

E. Clustering de locutores

El caso de clustering de locutores nos encontramos con un escenario bastante distinto a los anteriores. No disponemos de un fondo de datos grande para entrenar un modelo, todo se realiza con los mismos datos de trabajo. Esto significa que tenemos muy pocos datos para entrenar las GMM. En los primeros intentos de desarrollar el modelo nos encontramos con que las puntuaciones de los audios con los audios eran muy bajas, del orden de e-9, esto se debe a que usábamos un modelo con 12 coeficientes MFCC y 6 Gaussianas. Esto causaba que las GMM no llegaran a converger y den puntuaciones tan bajas a los audios. Aún y dar estos resultados tan

malos en las scores de las GMM las métricas finales no son especialmente malas.

```
Precision: 0.804569783803234 Recall: 0.7453565789443343
```

Fig. 13. Resultados 12 MFCCs 6 gaussianas

Para mejorar esto hicimos varias pruebas y los mejores resultados se consiguen usando 6 coeficientes MFCC y solo 2 GMMs. Al usar estos parámetros las logscore son de valores más razonables y las métricas de evaluación tienen mejores resultados.

```
Precision: 0.804569783803234 Recall: 0.7453565789443343
```

Fig. 14. Resultados 6 MFCCs 2 gaussianas

X. Conclusiones del trabajo

Para la creación del modelo de GMM de cada uno de los locutores es necesario utilizar gran cantidad de audios, los cuales después se deben de caracterizar en un proceso que requiere de muchos recursos computacionales. Por lo que es un proceso costoso temporal y economico (conseguir los audios suele requerir pagar a gente por ello).
Es muy importante utilizar distintos archivos para entrenar el sistema y para la evaluación, ya que se quiere conseguir un sistema que sea capaz de generalizar y eso solo se puede comprobar haciendo tests con audios que el sistema no ha visto. La cantidad de coeficientes y Gaussianas debe ser lo suficientemente grande como para caracterizar bien las señales pero sin llegar a provocar `overfitting`, es decir que la caracterización sea igual a la señal original.
Existen herramientas como la librería SPTK, Speech Toolkit que nos facilitan mucho el trabajo pero que requieren de conocimientos de `bash` para poder automatizarlo todo.

References

[1] Speech Signal Processing. SPTK, Speech Toolkit. Mayo 2019. http://sp-tk.sourceforge.net/

[2] Practical Cryptography. Mel Frequency Cepstral Coefficient (MFCC). Mayo 2019. http://notedetection.weebly.com/center-clipping.html