MANUAL DE APOYO PARA DIARIZACIÓN DE LOCUTORES EN PYBK Y S4D

Diarización de locutores

Por medio de este documento se recogen recomendaciones que se han obtenido tras la utilización de los sistemas de diarización pyBK y S4D. Se puede disponer de la información y el material necesario para su implementación, realizada por sus respectivos autores, en los repositorios de GitHub correspondientes. Se ofrece el enlace a los repositorios de cada sistema como nota a pie de página de cada método, en donde se puede disponer del material que los autores ofrecen para cada método. La última versión de los repositorios a los que se ha accedido ha sido en febrero de 2021.

Adicionalmente, se describen los bloques de Eliminación de silencios y de Evaluación de los resultados, así como el procedimiento para procesar los ficheros de audio con la información obtenida tras la diarización

1. Introducción

Una buena manera de comenzar con los sistemas de diarización y como toma de contacto con algunos de los sistemas de diarización existentes, es mediante el siguiente enlace:

https://github.com/wq2012/awesome-diarization

En esa página se pueden encontrar enlaces a repositorios de algunos sistemas de diarización (*Software -> Framework*), en los que se pueden encontrar scripts necesarios para su implementación y explicación sobre su funcionamiento. Además, se van actualizando, por lo que se dispondrá de las últimas versiones en el momento que se acceda.

En esa página también hay contenido extra para el proceso completo de diarización: *datasets* para pruebas, sistemas para evaluar resultados, enlaces a artículos sobre funcionamiento y evaluación de los sistemas. Especialmente útiles son algunos de los blogs y videotutoriales que se ofrecen en la página, algunos más específicos y otros más introductorios a la diarización[1].

2. Eliminación de silencios[2]

Este paso es opcional y en el trabajo se ha realizado previamente a ejecutar los sistemas de diarización. Muchos de los sistemas de diarización también incluyen una etapa para detectar habla y eliminar silencios. Si se dispone de audios en buenas condiciones, se puede saltar al paso siguiente.

En la carpeta Silence_Removal, encontramos los scripts: *silenceremove.py* y *silenceremove_all.py.*

silenceremove.py: Realiza la eliminación de silencios para un fichero de audio y guarda el fichero procesado en la ruta establecida.

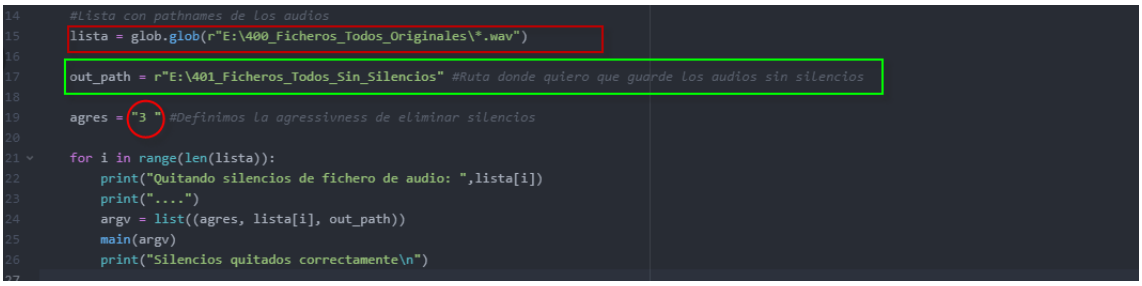
silenceremove_all.py: Script general que llama a *silenceremove* por cada fichero de audio de entrada.

[1]https://medium.com/saarthi-ai/who-spoke-when-build-your-own-speaker-diarization-module-from-scratch-e7d725ee279 Saxena, Rahul

[2]https://ngbala6.medium.com/audio-processing-and-remove-silence-using-python-a7fe1552007a Murugan, Bala

A configurar:

- Ruta de ficheros de entrada (**rojo**)
- Definir nivel de “agresividad” (entre 0 y 3), para indicar lo estricto que sea quitando silencios. Por defecto, al máximo: 3
- Ruta de ficheros de salida (ficheros sin silencios) (**verde**)



Por defecto, en silenceremove.py se ha definido que los ficheros de audio de salida tengan añadido en su nombre “Non-Silenced-Audio”.

Requerimientos: *webrtcvad == 2.0.10*

3. Sistemas

En los siguientes apartados se describen cómo utilizar los sistemas de diarización utilizados. Se describen las modificaciones realizadas y una descripción de sus parámetros para facilitar su configuración. Para cada sistema, se ofrece el enlace al repositorio de github donde se encuentra la implementación original.

En primer lugar, se debe tener en cuenta que la manera de utilizarlos será pasando las rutas de las carpetas o ficheros tanto de entrada como de salida y estableciendo los parámetros que se requieran en cada momento.

3.1 pyBK[3]

La configuración de pyBK se hace desde un fichero de configuración de configparser[4]. En resumen, en este fichero establecemos tanto las rutas de entrada/salida de los ficheros de audio como los parámetros característicos de la diarización con binary keys (número de MFCCs, tamaño del KBM, tipo de agrupamiento…). Es un fichero *.ini* pero que se puede editar como un *txt*. Para ver los diferentes parámetros de manera más fácil y visual se puede ver el pdf *pyBKconfig.pdf.*

Una de las características de este tipo de ficheros es que se divide en secciones y en cada sección tiene sus llaves y el valor asociado. Es parecido a un diccionario de Python.

```
[SEGMENT]
# This section configures the frames of features on which the binary keys/cumulative vectors are extracted
# Window size in frames
length = 100
# Window increment after and before window in frames
increment = 100
# Window shifting in frames
rate = 100
```

Importante: No cambiar nombres de secciones o valores de las llaves, ya que son las que utiliza el código para buscarlas. Si se cambian, cambiar en código también.

Establecer configuración

Abrimos el fichero *.ini* y establecemos los parámetros que se desean. Se han hecho algunas modificaciones para establecer una ruta de salida para los ficheros de audio del periodista. Principalmente, en las rutas indicamos la ruta donde están los ficheros de audio que se van a diarizar (**1**), la ruta para los ficheros de audio con la voz del periodista (**2**) y la ruta para los ficheros rttm (**3**):

```
[PATH]
# Path to the respective necessary files
# Audio files, necessary if performFeatureExtraction=1
1 audio = C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\audio_default\
#Features in HTK format, necessary if performFeatureExtraction=0
features = ./features/
# UEM files indicate the audio part to be considered in an audio file basis. The system expects a .uem file with the same name as its audio file
UEM = ./uem/
# SAD files indicate the speech part to be considered in an audio file basis. The system expects a .lbl/.mdtm/.rttm file with the same name as its audio file
#Audio periodista output. Le indicamos donde guarda los audios con la parte del periodista
2 audio_per = C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\out\
SAD = ./sad/
# Diarization output folder will contain the concatenated files diarization outputs in .mdtm/.rttm format
3 output = C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\out\
```

Importante: Poner la última "\" del final de cada ruta.

[3] https://github.com/josepatino/pyBK
[4] Documentación de configparser: https://docs.python.org/3/library/configparser.html

El resto de campos, tienen el mismo formato que los que vienen en el *.ini* de pyBK, que incluyen una descripción en cada etapa. A modo de resumen, es recomendable los siguientes valores en estos parámetros:

useRelativeKBMsize = 1. Es mejor que el número de gaussianas del KBM sea proporcional al tamaño de los ficheros de entrada. Normalmente, los ficheros con los que se van a trabajar tendrán duraciones similares, pero es recomendable no dejar un valor fijo para este parámetro, en caso de que haya ficheros concretos que sean demasiado cortos o largos respecto del resto.

metric = cosine. Es la métrica utilizada para realizar el cálculo de las distancias entre vectores acumulativos de manera más rápida. Su elección es algo más teórica, pero es una de las mejoras propuestas sobre el sistema original, por lo que es recomendable su elección.

modelSize. Si se decide aplicar resegmentación (que es recomendable), un valor alto de este parámetro puede hacer que el sistema tarde mucho en hacer el proceso completo de diarización sin producirse una mejora de los resultados.

Procesado posterior pyBK

Se ha añadido una función de post – procesado tras la diarización (*procesapyBK*) con dos tareas principales:

Separar los rttms. Por defecto, pyBK guarda los resultados de la diarización en un único fichero rttm para todos los ficheros. El nombre del fichero es el que se ha establecido en el campo *name* de la sección [EXPERIMENT]. En *procesapyBK* se separan los resultados de diarización de N ficheros en N rttms diferentes.

Crear audio del periodista: Se realiza con la función *cortawav (*se explica posteriormente*).* Se utiliza la información obtenida tras la diarización para identificar los intervalos de tiempo del periodista y crear el audio de salida que contenga su voz.

procesa_pyBK (*rttm_juntos_filepath, rttm_separados_path, audio_entrada_path, audio_periodista_path*)

rttm_juntos_filepath: ruta del fichero donde pyBK guarda los resultados (todos los rttm juntos)

rttm_separados_path: ruta donde guardar los rttms de cada fichero separados

audio_entrada_path: ruta de los audios de entrada

audio_periodista_path: ruta del audio con el periodista

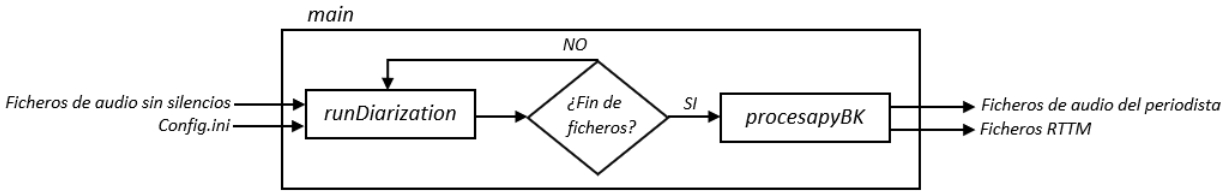
Importante: Por defecto, si se ejecuta *python main.py*, la configuración utilizada es la definida en configFile del código:

```
115     if len(sys.argv) >= 2:
116         configFile = sys.argv[1]
117     else:
118         configFile = r"C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\config_DIHARD_PathAdaptado.ini"
119     config = configparser.ConfigParser()
120     config.read(configFile)
```

Si se dispone de varias configuraciones diferentes, se pueden ejecutar fácilmente poniendo el nombre del *.ini* tras el main.py al ejecutar:

```
User@LAPTOP-IH5DB9PF MINGW64 ~/MASTER/TFM/31_pyBK_Completo/pyBK-master
$ python main.py config_1.ini
```

Importante: Ejecutar como administrador, si no, puede haber problemas para acceder a los ficheros.



Solución a errores pyBK

Durante la implementación y prueba de pyBK, se han producido algunos errores. Se indica el error, en qué parte se ha producido, su motivo, y la solución.

Overflow Error: cannot convert float infinity to integer

```
Traceback (most recent call last):
  File "main.py", line 140, in <module>
    runDiarization(os.path.splitext(os.path.basename(showName))[0],config)
  File "main.py", line 65, in runDiarization
    kbmSize = int(np.floor(poolSize*config.getfloat('KBM','relKBMsize')))
OverflowError: cannot convert float infinity to integer
```

Se ha producido en el cálculo del tamaño de KBM (*kbmsize*). Es debido a que en el fichero de configuración se ha establecido un número de gaussianas mínimo demasiado alto para el fichero, que es de menor duración.

Debido a que el audio es corto, al dividir el tamaño entre el número de gaussianas mínimas para calcular el *windowRate*, da un valor de 0. Por tanto, cuando intenta calcular el número de gaussianas totales, dividiendo las características acústicas entre cada cuantas ventanas se escogen, hace una división entre 0, que da como resultado ∞. Eso da el error de que el *kbmSize* no pueda ser infinito (espera *float* pero llega infinito).

```
56      #create the KBM
57      print('Training the KBM... ')
58      #set the window rate in order to obtain "minimumNumberOfInitialGaussians" gaussians
59      if np.floor((nSpeechFeatures-config.getint('KBM','windowLength'))/config.getint('KBM','minimumNumberOfInitialGaussians')) < config.getint('KBM','maximumKBMWindowRate'):
60          windowRate = int(np.floor((np.size(data,0)-config.getint('KBM','windowLength'))/config.getint('KBM','minimumNumberOfInitialGaussians')))
61      else:
62          windowRate = int(config.getint('KBM','maximumKBMWindowRate'))
63      poolSize = np.floor((nSpeechFeatures-config.getint('KBM','windowLength'))/windowRate)
64      if  config.getint('KBM','useRelativeKBMsize'):
65          kbmSize = int(np.floor(poolSize*config.getfloat('KBM','relKBMsize')))
66      else:
67          kbmSize = int(config.getint('KBM','kbmSize'))
```

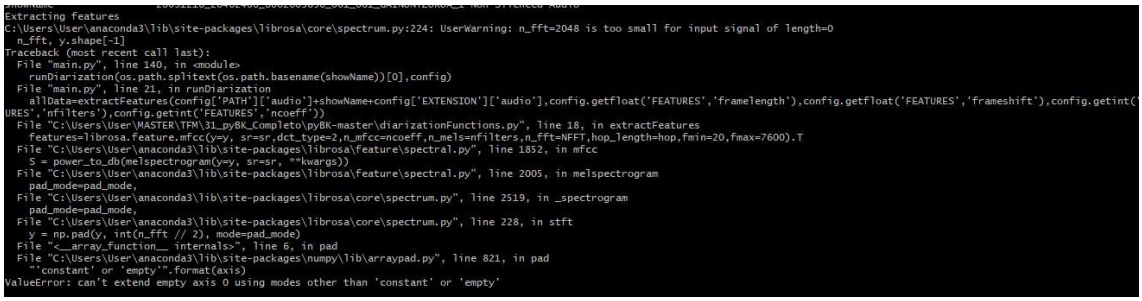
Index Error: index 0 is out of bounds for axis 0 with size 0

```
Selecting best clustering...
Traceback (most recent call last):
  File "main.py", line 140, in <module>
    runDiarization(os.path.splitext(os.path.basename(showName))[0],config)
  File "main.py", line 84, in runDiarization
    bestClusteringID = getBestClustering(config['CLUSTERING_SELECTION']['metric_clusteringSelection'], segmentBKTable, segmentCVTable, finalClusteringTable, k, config.getint('CLUSTERING_SELECT
ION','maxNrSpeakers'))
  File "C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\diarizationFunctions.py", line 519, in getBestClustering
    bestClusteringID = np.maximum(np.where(nrSpeakersPerSolution==np.maximum(np.minimum(maxNrSpeakers,np.max(nrSpeakersPerSolution))-1,1))[0][0],bestClusteringID)
IndexError: index 0 is out of bounds for axis 0 with size 0

User@LAPTOP-IH5DB9PF MINGW64 ~/MASTER/TFM/31_pyBK_Completo/pyBK-master
$
```

Se ha producido en la función *getBestClusteringID*. Ocurre si el valor de maxNrSpeakers es excesivamente alto, que hace que se detecte como máximo un índice fuera de rango. Ajustando el valor de maxNrSpeakers se soluciona, que depende del tipo de audio, por lo que no hay un valor fijo, aunque mejor que sea bajo.

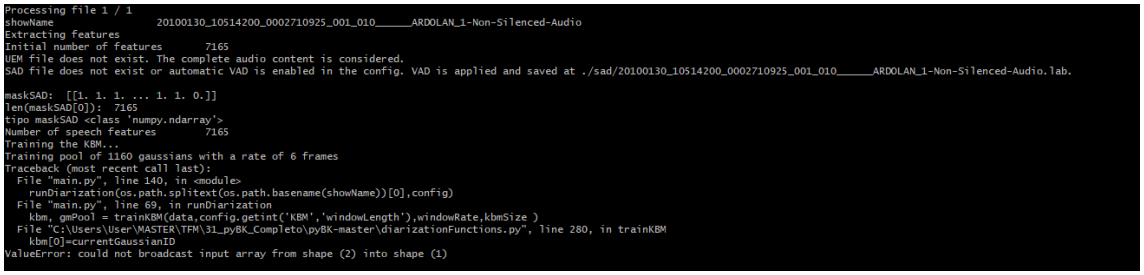
Error: ValueError: can't extend empty axis 0 using modes other than 'constant' or 'empty'



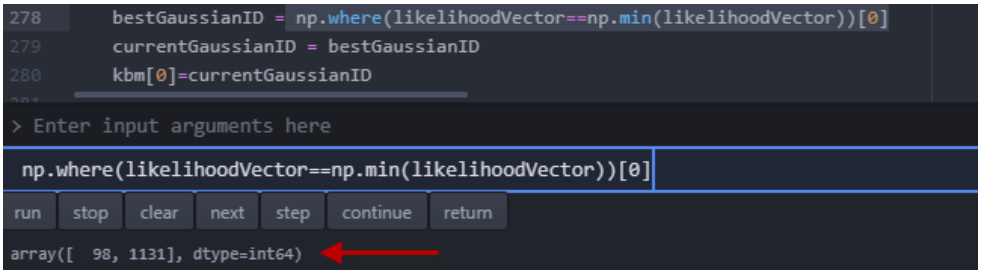
Es posible que se deba a que el audio sea de duración 0, porque inicialmente venía así o tras quitarle silencios.

Importante: antes de hacer pyBK de forma masiva, quitar audios "problemáticos" (audios de duración 0).

Error: ValueError: could not broadcast input array from shape (2) into shape (1)



Este error se produce entrenando el KBM porque cuando calcula las distancias entre gaussianas para determinar la que escoge para añadir al KBM, hay dos gaussianas que tienen el valor mínimo, en vez de una. Por tanto, en vez de haber una única posición, hay dos (o más, si hay más de dos posiciones con el mismo valor), y da un error de dimensiones al poner en un array de 1 elemento *kbm [0]* más de un elemento.



Se hace una comprobación de si el número de posiciones es mayor que uno, y se escoge la primera:

```
    # Define the global dissimilarity vector
    v_dist = np.inf*np.ones((numberOfComponents,1))
    # Create the kbm itself, which is a vector of kbmSize size, and contains the gaussian IDs of the components
    kbm = np.zeros((kbmSize,1));
    # As the stored likelihoods are negative, get the minimum likelihood
    bestGaussianID = np.where(likelihoodVector==np.min(likelihoodVector))[0]
    if bestGaussianID.shape > (1,):
        bestGaussianID = np.array([bestGaussianID[0]])#Porque hay casos en los que el valor mínimo está en dos posiciones
        print("Solucion error de GaussianID")
    currentGaussianID = bestGaussianID
    kbm[0]=currentGaussianID
    v_dist[currentGaussianID]=-np.inf
    # Compare the current gaussian with the remaining ones
    dpairsAll = cdist(muVector,muVector,metric='cosine')
    np.fill_diagonal(dpairsAll,-np.inf)
```

Lo mismo cuando calcule las gaussianas con mayor diferencia:

```
            currentGaussianID = np.where(v_dist==np.max(v_dist))[0]
            if currentGaussianID.shape > (1,):
                currentGaussianID = np.array([currentGaussianID[0]])
            kbm[j]=currentGaussianID
            v_dist[currentGaussianID]=-np.inf
        return [kbm, gmPool]
```

ValueError: Fitting the mixture model failed because some components have ill-defined empirical covariance (for instance caused by singleton or collapsed samples). Try to decrease the number of components, or increase reg_covar.

```
Performing GMM-ML resegmentation...
Traceback (most recent call last):
  File "main.py", line 140, in <module>
    runDiarization(os.path.splitext(os.path.basename(showName))[0],config)
  File "main.py", line 93, in runDiarization
    finalClusteringTableResegmentation,finalSegmentTable = performResegmentation(data,speechMapping, mask,finalClusteringTable[:,bestClusteringID.astype(int)-1],segmentTable,config.getint('RES
EGMENTATION','modelSize'),config.getint('RESEGMENTATION','nbIter'),config.getint('RESEGMENTATION','smoothWin'),nSpeechFeatures)
  File "C:\Users\User\MASTER\TFM\31_pyBK_Completo\pyBK-master\diarizationFunctions.py", line 602, in performResegmentation
    gmm.fit(data[spkIdxs,:])
  File "C:\Users\User\anaconda3\lib\site-packages\sklearn\mixture\base.py", line 191, in fit
    self.fit_predict(X, y)
  File "C:\Users\User\anaconda3\lib\site-packages\sklearn\mixture\base.py", line 234, in fit_predict
    self._initialize_parameters(X, random_state)
  File "C:\Users\User\anaconda3\lib\site-packages\sklearn\mixture\base.py", line 155, in _initialize_parameters
    self._initialize(X, resp)
  File "C:\Users\User\anaconda3\lib\site-packages\sklearn\mixture\gaussian_mixture.py", line 649, in _initialize
    covariances, self.covariance_type)
  File "C:\Users\User\anaconda3\lib\site-packages\sklearn\mixture\gaussian_mixture.py", line 333, in _compute_precision_cholesky
    raise ValueError(estimate_precision_error_message)
ValueError: Fitting the mixture model failed because some components have ill-defined empirical covariance (for instance caused by singleton or collapsed samples). Try to decrease the number o
f components, or increase reg_covar.
```

Error producido en la parte de resegmentación, debido a que el valor de GMMSize sea demasiado alto para algunos audios. La manera más sencilla de solucionarlos es aumentando el valor de reg_covar (que por defecto está definido a 1e-6). Es un valor que regulariza los valores en la matriz de covarianzas para asegurar que los valores sean positivos.

Se puede ajustar en *diarizationfunctions.py* en la función *performResegmentation* en *gmm=mixture.GaussianMixture,* y se establece un valor superior a 1e-6 (con 1e-5 - 1e-4 ha funcionado, pero se puede ajustar más).

Requerimientos: *numpy = 1.18.1, scipy = 1.4.1, scikit-learn = 0.24.1, librosa = 0.8.0, webrtcvad = 2.0.10*

3.2 Sistema s4d[5]

Una vez tenemos los ficheros con los silencios quitados, que son los que vamos a utilizar para la diarización, abrimos el script ***s4d_completo_main.py***. En las primeras líneas del script definimos las rutas de entrada y las de salida y los parámetros que queremos para la configuración:

Rutas de entrada (**verde**): ruta de los ficheros sin silencios.

Rutas de salida(**rojo**):

audio_edit_out: Ruta de salida para los audios que contengan la voz del periodista(*wav*)

rttm_out: Ruta de salida para los ficheros rttm obtenidos con el sistema (*txt*)

logs_out: Ruta de salida para los logs de cada etapa (*txt*)

```
if __name__ == '__main__':
    t_ini_todos = time()#Calculamos el tiempo que tarda
    input_path = sys.argv[1:]
    print("Argumentos de entrada: ",  sys.argv[1:])
    lista = glob.glob(r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\001_Ficheros_Non_Silences\*.wav")

    audio_edit_out = r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\a"
    rttm_out = r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\a"
    logs_out = r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\a"

    #Definimos los parametros para la diarizacion
    #Threshold for: * linear segmentation (step 3) * BIC HAC (step 4) * Vitterbi (step 5)
    thr_l = 1 #Defecto 2
    thr_h = 2.5#Defecto 3
    thr_vit = -250 #Defecto -250

    tam_ventana = 125  #Defecto 250.  Size of left or right windows (step 2)
```

Los parámetros que queremos en la segmentación, agrupamiento, resgementación, para poder definir lo estricto que va a ser determinando cambios de locutor en la segmentación y grupos diferentes en el agrupamiento.

```
    #Definimos los parametros para la diarizacion
    #Threshold for: * linear segmentation (step 3) * BIC HAC (step 4) * Vitterbi (step 5)
    thr_l = 1 #Defecto 2
    thr_h = 2#Defecto 3
    thr_vit = -250 #Defecto -250

    tam_ventana = 250  #Defecto 250.  Size of left or right windows (step 2)
```

***thr_l*:** umbral para la segmentación;

thr_h: umbral para el agrupamiento;

thr_vit: umbral para la resegmentación;

tam_ventana: tamaño de ventana para extracción de características

Cuanto menor sea el valor de ***thr_l*,** menos restrictivo es determinando cambios de locutor, lo que favorece que se detecten más cambios de locutor en la segmentación.

Cuanto menor sea el valor de ***thr_h***, menos restritivo es identificando dos grupos como grupos diferentes, lo que supone que en el resultado final haya un mayor número de grupos independientes / locutores.

[5] https://projets-lium.univ-lemans.fr/s4d/

El valor de ***thr_vit*** que viene por defecto (-250) ofrece buenos resultados.

El valor de ***tam_ventana*** es el tamaño de la ventana sobre el que se calcularán los coeficientes MFCC y demás parámetros para las características acústicas. Se puede reducir, pero hace que aumente el tiempo de ejecución y una reducción importante tampoco supone una mejora de resultados.

Los parámetros para la extracción de características se pueden modificar desde la carpeta s4d, en el script *utils.py* en *get_feature_extractor*.

```
    elif type_feature_extractor == 'basic':
        fe = FeaturesExtractor(audio_filename_structure=audio_filename_structure,
                               feature_filename_structure=None,
                               sampling_frequency=16000,
                               lower_frequency=133.3333,
                               higher_frequency=6855.4976,
                               filter_bank="log",
                               filter_bank_size=40, #Por defecto:filter_bank_size=40
                               window_size=0.025, #Por defecto: window_size=0.025
                               shift=0.01, #Por defecto:0.01
                               ceps_number=19, #Por defecto ceps_number=13
                               pre_emphasis=0.97,
                               keep_all_features=True,
                               vad=None,
                               save_param=["energy", "cep", "vad"]
                               )
```

Una cosa que se debe tener en cuenta es que se el código se va eliminando la información que extrae en formato HDF5 de *FeaturesExtractor* y que luego utiliza en *FeaturesServer*. Al usar el sistema en muchos audios, se produce error, y es información que tras diarizar el fichero no se vuelve a utilizar.

```

    #Diarizamos para todos los audios
    for i in range(len(lista)):
        if "KORTE" not in os.path.basename(lista[i]):
            print("Diarizando con s4d el fichero: ", os.path.basename(lista[i]))
            s4d_diar.diariza(lista[i], audio_edit_out, rttm_out, logs_out, thr_l, thr_h, thr_vit, tam_ventana)
            files_to_del =glob.glob(r"C:\Users\User\MASTER\TFM\39_3_s4d-master_Completo\out\audio\*")
            for i in range(len(files_to_del)):
                os.remove(files_to_del[i])
            print("Diarizacion completada")
            print("*************************************************\n\n")

```

Una vez definidos los parámetros y establecidas las rutas, sólo hay que ejecutarlo. El procesamiento completo (diarización + audio periodista + rttm + logs) se hace fichero a fichero. El procesamiento de diarización se hace en la función de s4d_diar ***diariza.***

diariza(*input_path, output_path_audio, output_path_rttm, output_path_log, umbral_l, umbral_h, umbral_vit, tam_ventana*)

input_path: Ruta de entrada del fichero que se va a diarizar

output_path_audio: Ruta de salida del fichero de audio procesado

output_path_rttm: Ruta de salida del rttm de la diarización del fichero

output_path_log: Ruta de salida de los logs de la diarización

umbral_l: Parámetro para ajustar la segmentación

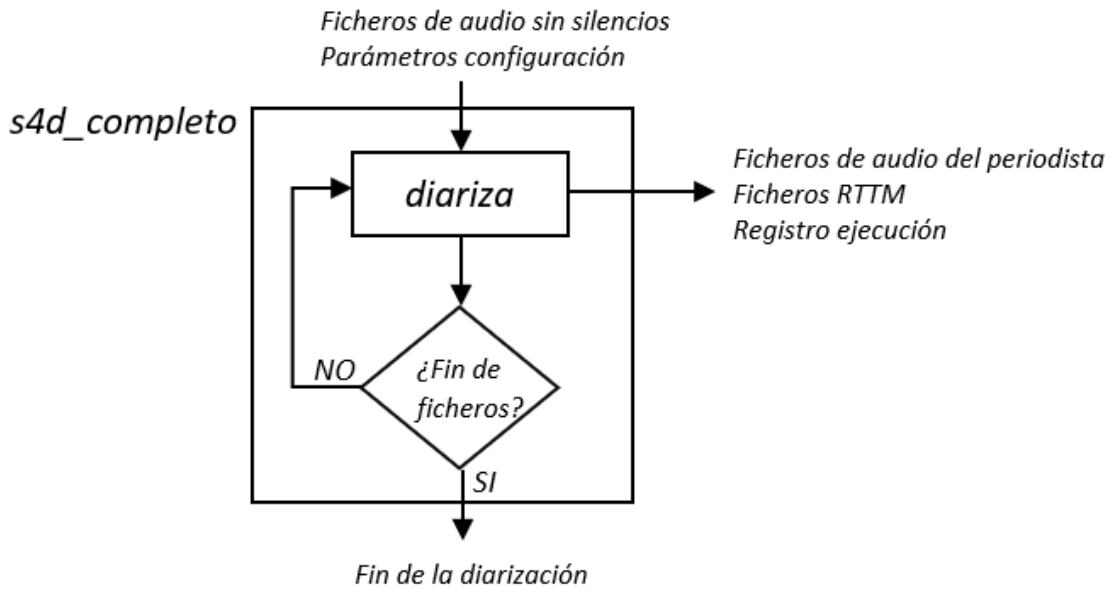
umbral_h: Parámetro para ajustar el agrupamiento

umbral_vit: Parámetro para ajustar la resegmentación

tam_ventana: parámetro para ajustar la ventana para extraer las características

En general, dejar un valor bajo para la segmentación para no perder cambios de locutores en esa etapa. En resegmentación, el valor por defecto (-250) está bien ajustado.

En los logs (registros de la ejecución), que son ficheros de texto, se pueden ver cuántos grupos hay en cada etapa y la duración de cada locutor.



Visualización de dendrogramas

Para visualizar los dendrogramas tras el agrupamiento en el proceso de diarización, descomentar la línea que llama a plot_dedrogram de matplotlib.pyplot.

```
128     link, data = plot_dendrogram(bic.merge, 0, size=(12,4), log=True) #Por defecto --> size=(25,6)
```

Requerimientos (por defecto): *pandas>=0.21.1, PyAudio>=0.2.11, numpy>=1.13.3, scipy>=0.19.0 matplotlib>=2.0.2, bottleneck>=1.3.1, setuptools>=38.5.2 sidekit>=1.3.6.5 six>=1.11.0, scikit_learn>=0.19.1, sortedcontainers>=1.5.9, h5py>=2.5.0*

Requerimientos (empleados): *numpy = 1.18.1, scipy = 1.4.1, matplotlib = 3.3.3, bottleneck = 1.3.2, setuptools = 45.2.0, sidekit = 1.3.6.9, six = 1.14.0, scikit-learn = 0.24.1, sortedcontainers = 2.1.0, h5py = 2.10.0*

3.3 Funciones auxiliares

Se han utilizado las siguientes funciones para el proceso de diarización completo:

corta_wav.py

corta_wav (*input_path, output_path, audio_times, identificador*)

Procesamiento del audio: información del fichero de audio y editarlos en función de los resultados de la diarización.

Parámetros:

input_path: Ruta del fichero de audio de entrada

output_path: Ruta del fichero de audio de salida

audio_times: Lista con inicio y final de los periodos en los que habla el locutor --> Formato: [tini, tfinal]

identificador: Identifica el sistema de diarización utilizado (string)

crea_rttm.py

Script con métodos para realizar procesamiento sobre los ficheros rttm

s4d_to_rttm (*input_path, output_path, data, channel*)

Parámetros:

input_path: Ruta del fichero RTTM de entrada

output_path: Ruta del fichero RTTM de salida (procesado)

data: Resultados de s4d. Por defecto, cada fila tiene 5 campos `que corresponden a cada turno de locutor: *[show, cluster, type, start, stop]*

channel: Número de canales

lab_to_rttm (*input_path, channel*)

Convierte el fichero .lab (formato de salida tras el marcado de tiempos en *wavesurfer*) en formato rttm.

Parámetros:

input_path: Ruta del fichero lab de entrada

channel: Número de canales

Por defecto, los ficheros resultantes se guardan en el mismo directorio que los. lab

4. Evaluación de los resultados

Los resultados de la diarización se evalúan en función del DER obtenido al comparar los ficheros RTTM obtenidos por los sistemas con los ficheros de referencia previamente obtenidos de forma manual por medio de alguna herramienta de procesamiento de audio (por ejemplo, wavesurfer).

La evaluación se realiza por medio del script del NIST md-eval-22.pl.

evalua_completo.py

Importante: Se ha establecido que calcule el DER de manera normal y también el DER específico de la parte del periodista. Si se desea únicamente el DER normal, dejar comentada la parte de código correspondiente a la comparación de periodista.

Definimos los siguientes parámetros/rutas:

ref_rttm_list: Ruta de los ficheros rttm de referencia

sis_rttm_list: Ruta de los ficheros rttm obtenidos tras la diarización

collar: Valor de collar utilizado. Por defecto 0.250

```
ref_rttm_list = glob.glob(r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\002_RTTM Referencia\21_Referencia Todos\*_ref_rttm.txt")
sis_rttm_list= glob.glob(r"E:\2_BASE_DE_DATOS_2\1_CASTELLANO\0_Seleccion_Diarizar\200_pyBK\8_config_elbow_512_KBM01\1_RTTM_Resultados\*rttmpyBK.txt")
collar = 0.250#Definirlo previamente
```

Los rttm de referencia mejor dejarlos en una única carpeta para no tener que cambiar la ruta cada vez que se cambie de sistema.

Los rttm del sistema definimos *rtmmpyBK.txt o *s4d_rttm.txt (o el identificador que se haya puesto). Así, únicamente se cambia ese identificador en cada evaluación.

Importante: Que tanto en la carpeta de los rttm de referencia como en la del sistema haya el mismo número de ficheros y que sean correspondientes a los mismos audios, para que las evaluaciones se hagan correctamente.

Para la parte de evaluación de periodista no hay que definir nada, la función ***evalua_periodista_RTTM*** se encarga de identificarlo y crear el rttm del periodista (se guarda en la misma carpeta que los del sistema)

Una manera cómoda para trabajar con los resultados es copiar la salida por pantalla y pegarla a un txt, para poder buscar por fichero. Se ha realizado una función ***logs_eval_visualizar*** que pasándole la ruta del .txt en el que se ha guardado los resultados por pantalla, guarda la información necesaria de la evaluación (directamente los errores y número de locutores detectados).