SYSTEME DE VERIFICATION DU LOCUTEUR

INTRODUCTION

Le but est de construire un système de vérification du locuteur.

Différentes étapes sont nécessaires à la construction et à la validation du système.

1) Nous utiliserons tout d'abord une **base de données** (ou corpus) utile à l'apprentissage du système et ensuite à son évaluation : pour cela, nous disposons de fichiers sons (ou signal).

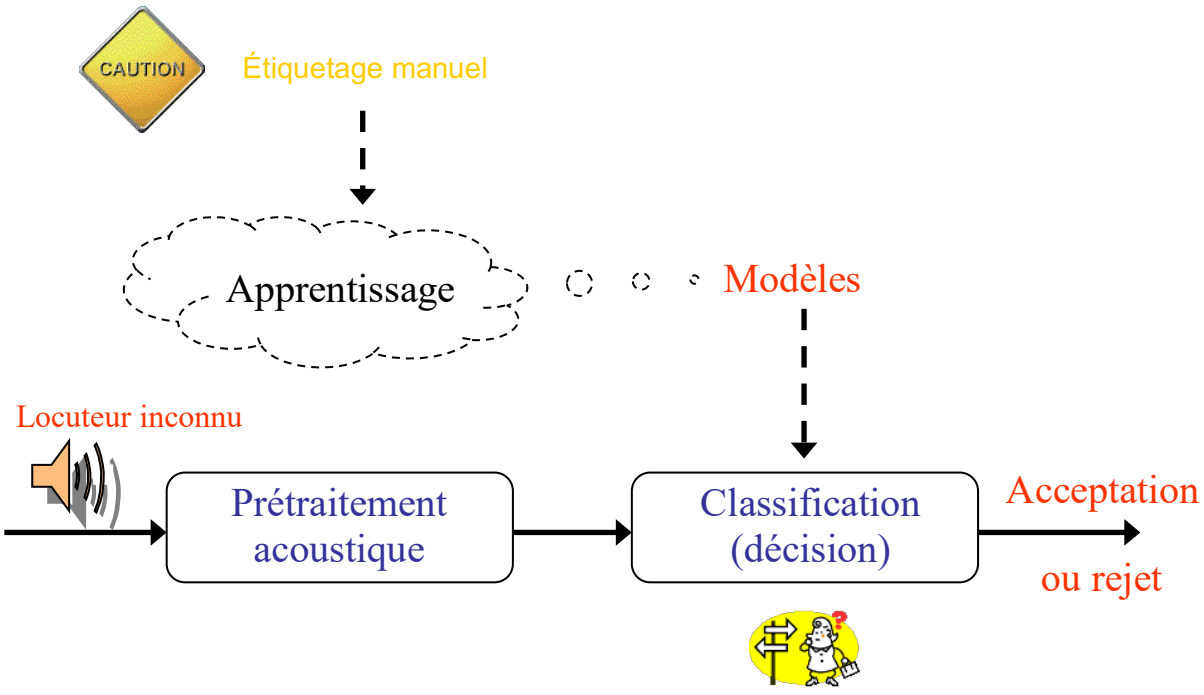
2) Une étape de **décomposition parole/non-parole** sera nécessaire afin de ne conserver que les zones de parole (correspondant aux locuteurs).

3) Puis, nous générerons les fichiers acoustiques correspondants : phase de **paramétrisation**.

4) Ensuite, nous ferons l'**apprentissage** des modèles de mélanges de lois gaussiennes : il s'agira de construire le modèle du « monde » et le modèle du locuteur cible.

5) Enfin nous passerons à la phase de **reconnaissance** (décision) : le locuteur testé, est-il le locuteur cible ou s'agit-il d'un imposteur ?

Voici, ci-dessous, le schéma général d'un système de vérification du locuteur.



1) PRESENTATION DE LA BASE DE DONNEES

La base de données est composée de 10 locuteurs francophones. Chaque locuteur possède 10 enregistrements (fichiers wave). La durée des fichiers varie de 13 à 34 secondes. Le découpage est le suivant : 8 enregistrements servent à l'apprentissage des modèles et 2 à la reconnaissance (tests).
Récupérer tous les enregistrements (fichiers .zip).

Le répertoire « APP » sert à l'apprentissage des modèles et le répertoire « RECO » sert à la reconnaissance (tests). Les fichiers sont nommés de la façon suivante : **L_x_fic_y.wav** avec **x** le numéro du locuteur et **y** le numéro du fichier.
Remarque : pour débuter les premières expérimentations pourront se faire sur un seul fichier, par exemple « L1_fic1.wav ».

Après avoir récupéré le programme Python « tp3.py », écrire une fonction « lecture » permettant de lire un fichier son (en normalisant les échantillons entre -1 et 1, grâce au nombre de bits de quantification), de connaître sa fréquence d'échantillonnage et sa durée. Le nombre de bits de quantification de nos fichiers est 16.
def lecture(fichier, nb_bits):

Rappel : utiliser la fonction « read » de *scipy.io.wavfile*.

2) DECOMPOSITION PAROLE/NON-PAROLE

Chaque fichier son n'est pas uniquement composé de parole ! En effet, des zones de silence et de bruit sont également présentes. Afin d'effectuer l'apprentissage des modèles (« monde » et « locuteur cible »), il est nécessaire d'isoler les zones de parole. Ce travail pourra s'appuyer sur le calcul de l'énergie à court terme. L'énergie est calculée sur des fenêtres de taille « taille_fenetre » avec un recouvrement de moitié. Classiquement la variable « taille_fenetre » est égale à 256, 512 ou 1024 points (échantillons).

Ecrire une fonction « etiquetage » (utilisant la fonction « energie » fournie) permettant d'étiqueter un fichier son en parole et non-parole. Le seuil est à fixer par vos soins.
def etiquetage(signal, taille_fenetre, seuil):

Remarques :

- *lire et comprendre la fonction « energie »* permettant de calculer l'énergie à court terme d'un signal en fonction de la taille des fenêtres d'analyse. Ce traitement a été vu lors de l'Introduction au Son en 1A.
- la sortie sera booléenne « 0 » (ou « False ») en l'absence de parole et de « 1 » (ou « True ») en présence de parole.

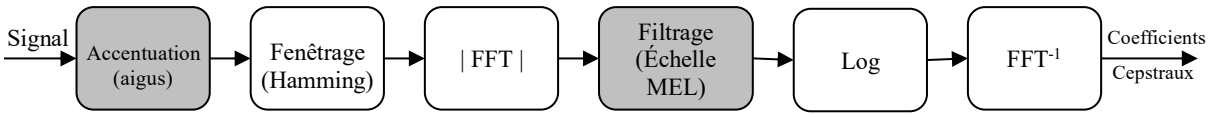
Ecrire une fonction « etiquetage_total » (utilisant la fonction « etiquetage ») permettant d'étiqueter en parole/non-parole tous les fichiers sonores du répertoire « APP ».
def etiquetage_total(nb_bits, taille_fenetre, nbe_loc, nbe_fic, seuil):

Les résultats de la fonction « etiquetage » seront enregistrés dans des fichiers texte (avec la fonction Numpy *savetxt*) portant le nom des fichiers son de départ avec l'extension « .lab » et seront rangés dans le répertoire « LABELS ». Ce répertoire sera au même niveau que le répertoires « APP » et « RECO ».

3) PARAMETRISATION

Cette phase correspond à la représentation acoustique du signal. Nous utilisons des coefficients cepstraux (MFCC). La taille de la fenêtre d'analyse « taille_fenetre » est la même que précédemment (cf. fonction « energie ») pour faciliter les traitements par la suite. Généralement le nombre de coefficients cepstraux « nbe_coef » est fixé à 8, 12, 16, …, 30.

Voici le schéma classique d'extraction de ces paramètres :



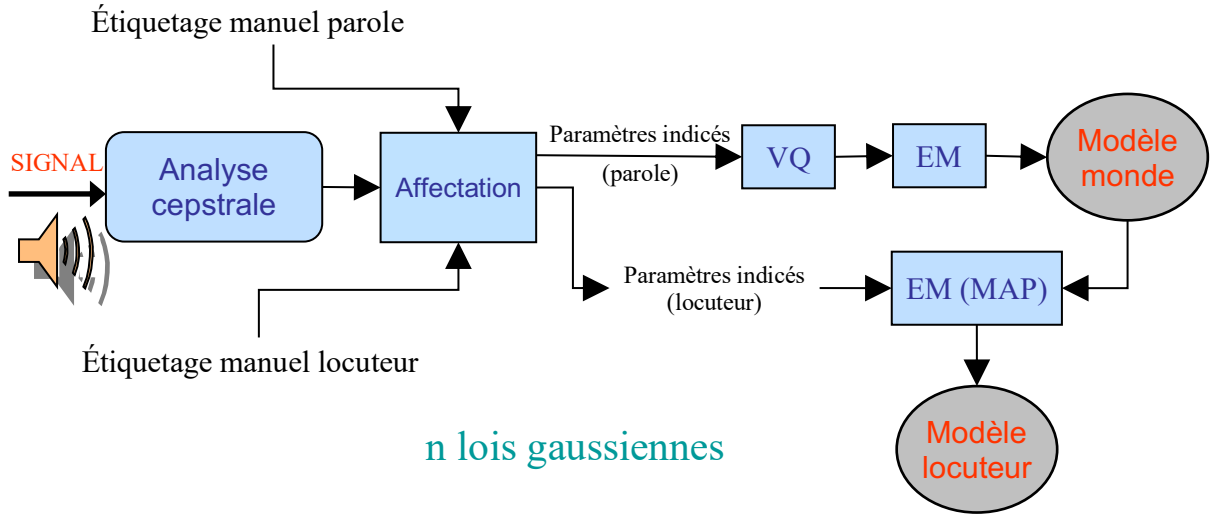
Lire et comprendre la fonction « parametrisation ». Là aussi, ce traitement a été vu lors de l'Introduction au Son en 1A. Le résultat de cette paramétrisation est un fichier composé d'une suite de valeurs telles : **$Coef_1$ $Coef_2$ … $Coef_{nbe\_coef}$**

Écrire une fonction « parametrisation_total.m » (utilisant la fonction « parametrisation » fournie) permettant de calculer les MFCC pour tous les fichiers du répertoire « APP ».
def parametrisation_total(nb_bits, taille_fenetre, nbe_coef, nbe_loc, nbe_fic):

Les résultats de la fonction « parametrisation » seront enregistrés dans des fichiers texte portant le nom des fichiers son de départ avec l'extension « .mfcc » et seront rangés dans le répertoire « MFCC ».

4) APPRENTISSAGE

Voici le schéma général permettant de créer les modèles du monde et du locuteur cible.



Afin d'effectuer l'apprentissage, il convient d'utiliser les vecteurs MFCC correspondant aux modèles. Pour cela, lors d'une phase d'affectation, les étiquetages en parole et en locuteur vont nous permettre de choisir les paramètres associés à chacun des modèles.

Utiliser et comprendre la fonction « affectation » permettant de créer deux fichiers contenant l'ensemble des paramètres (MFCC) indicés nécessaire à l'apprentissage des modèles du « monde » et du « locuteur ». Le choix (numéro) du locuteur est passé en paramètre.

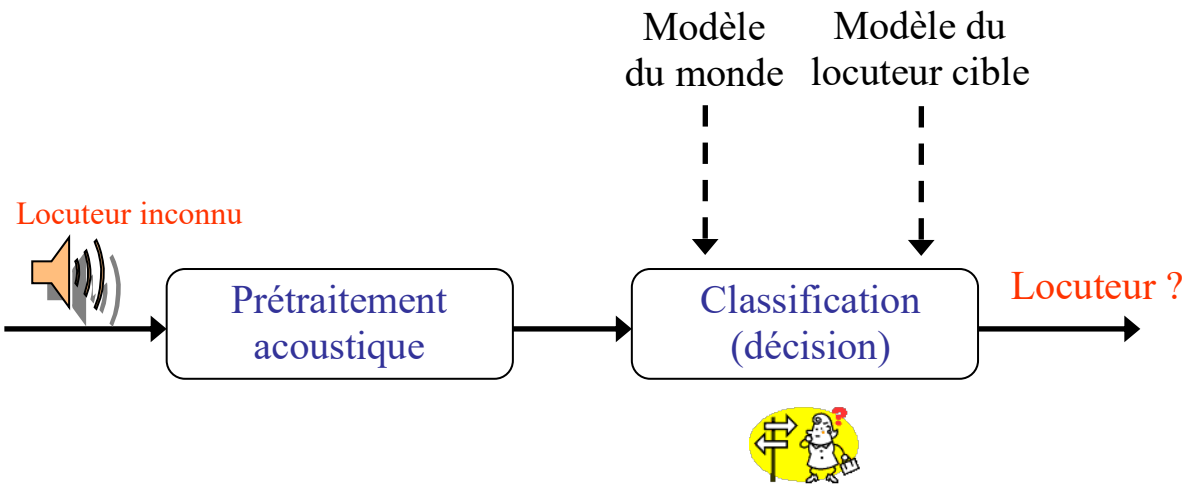
La première étape de l'apprentissage des modèles correspond à l'initialisation du modèle du « monde ». En général, une quantification vectorielle (VQ) est réalisée sur l'ensemble des paramètres (MFCC) correspondant à de la parole. Nous utiliserons, lors de ce TP, l'algorithme des « k-means ». Le nombre de centres (ou nombre de gaussiennes) est fixé à une puissance de 2 : nous prendrons ici 1, 2, 4, 8, ou 16.

Nous nous servirons de la fonction « GaussianMixture » de *sklearn.mixture* pour faire l'apprentissage des modèles du monde et du locuteur choisi. Cette fonction fait appel à l'algorithme des k-means (VQ) pour initialiser les modèles et elle fait ensuite la ré-estimation des modèles en utilisant l'algorithme d'optimisation EM.

Faire 2 appels à la fonction nommée « em » dans « tp3.py ». Vous utiliserez la méthode « fit » afin de créer les 2 modèles (« monde » et « Lx »). Pour plus de détails, voir :
https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture
Cette fonction « em » utilisera les vecteurs de données à traiter (paramètres indicés parole dans un cas et locuteur dans l'autre cas) ainsi que le nombre de lois gaussiennes. Elle fournira en sortie les modèles du monde et du locuteur représentés par les poids (.weights_), les moyennes (.means_) et les variances (.covariances_) de chacune des lois gaussiennes.

5) RECONNAISSANCE

Voici le schéma général permettant d'effectuer la reconnaissance.



Lors de cette phase, il convient de calculer les vecteurs MFCC, issus du prétraitement acoustique, du locuteur inconnu : pour cela, vous utiliserez les 20 fichiers sons contenus dans le répertoire « RECO ». Vous utiliserez également les modèles issus de l'apprentissage : « monde » et « Lx ». Votre système fonctionnera parfaitement si les 2 fichiers du locuteur choisi comme étant le locuteur cible, sont acceptés par le système et que les 18 autres sont rejetés.

Nous utiliserons la méthode « score_samples » de « GaussianMixture » afin de calculer la vraisemblance (probabilité) des vecteurs acoustiques par rapport aux 2 modèles. Nous considèrerons que la décision (acceptation/rejet du locuteur par fichier) s'effectue par un vote

(à la majorité) et que le score de confiance correspond au rapport entre le nombre de vecteurs acoustiques associés au locuteur et le nombre total.

Ecrire la fonction « tests_total » permettant d'effectuer la reconnaissance des fichiers sons inconnus situés dans le répertoire « RECO ». Cette fonction prendra en entrée la taille de la fenêtre d'analyse, le nombre de coefficients cepstraux ainsi que les modèles, composés de mélanges de lois gaussiennes. Elle fournira en sortie le taux de reconnaissance de chacun des fichiers.

```
def tests_total(nbe_loc, nbe_fic, nb_bits, taille_fenetre, nbe_coef, monde, loc, seuil)
```

Une première amélioration de votre système consistera à n'**effectuer la classification que des zones de parole** de chaque fichier son inconnu : la phase d'étiquetage sera alors nécessaire pour isoler les paramètres acoustiques correspondant à de la parole.

Vous pourrez tester et valider votre système, par exemple, en choisissant un autre locuteur cible.

Vous pourrez modifier, améliorer vos résultats en faisant varier l'ensemble des constantes/valeurs que vous avez utilisé pour construire ce système de vérification du locuteur, notamment :

- le nombre de lois gaussiennes : n,
- le nombre de paramètres (coefficients cepstraux) : nbe_coef,
- la taille de la fenêtre d'analyse : taille_fenetre,
- le seuil de l'énergie,
- le type de paramètres,
- le nombre de fichiers d'apprentissage et de tests.