Activité 3 : Reconnaissance de la parole

(MFCC, DTW)

Le travail à rendre contiendra le fichier .py, et l'analyse de la reconnaissance (.odt, .doc ou .pdf), le tout compressé dans un fichier .zip.
Il est possible de se mettre par binôme, mais constitué lors des séances

Objectifs : Réaliser un petit système de reconnaissance de la parole permettant de discriminer le OUI du NON.

Outils : Python et Winsnoori pour lire et visualiser le signal modifié.

Nous allons calculer les coefficients MFCC qui serviront ensuite comme représentation pour réaliser la reconnaissance de la parole.

Enregistrer un fichier son (à faire pour la prochaine séance)

Le but est de d'obtenir des fichiers de test avec votre voix. Plusieurs outils sont possibles pour enregistrer une voix. Quel que soit ce que vous utiliserez vous devez obtenir un fichier au format WAV à **22050Hz, 16bits, mono**. Voici 2 possibilités mais vous pouvez utiliser autre chose :

- *Audacity avec les paramètres suivants :*
  - Pistes/Ajouter nouvelle/Piste Mono
  - Projet à 22050
  - Format d'échantillonnage : 16bits PCM
  - Fréquence d'échantillonnage : 22050 Hz
  - MAJ+enregistrer pour enregistrer sur cette nouvelle piste vierge
  - Sélectionnez le signal du début à la fin du OUI/NON.
  - Exporter
- *WinSnoori*
  - Menu File/Record
  - Choisir 1s

Vous enregistrerez 1 OUI et 1 NON par étudiant.

Puis les sons suivants (1 par fichier, et par étudiant)
NUIT, BUIS, BON, DON, ZÉRO, UN.

Extraire les coefficients MFCC

Réalisez un programme python qui calcule les coefficients MFCC pour un signal de parole. Le programme prendra en entrée un fichier .wav et génèrera un fichier .mfcc.

Pour calculer les coefficients MFCCs vous utiliserez la librairie Librosa (assez utilisée en traitement du signal : `conda install -c conda-forge librosa` ou `pip install librosa`). Les pages d'aide pour librosa.feature.mfcc et librosa.feature.delta seront utiles pour réaliser le fichier .mfcc avec la configuration suivante :

win_length = 25 * sample_rate // 1000 # = 25 ms = length of a time frame
hop_length = 10 * sample_rate // 1000 # = 10 ms = frame periodicity
n_mfcc = 13 # Number of MFCC coeffs
window= tableau contenant la fenêtre de hamming

Le but est de calculer les 13 premiers coefficients et leur delta (dérivée) et delta-delta (accélération). Et de ne garder dans le fichier final que les 12 coefficients (1 à 12) en enlevant le premier (numéro 0) contenant principalement l'information de l'énergie. Nous aurons au final un vecteur MFCC de 12x3 = 36 coefficients.

Pour faciliter la lecture, le fichier .mffcc sera du texte avec le format suivant :

```
Nombre de vecteurs
Vecteur 1 : 36 valeurs flottantes
Vecteur 2 : 36 valeurs flottantes
…
```

Réalisez les fichiers .mfcc pour l'ensemble des fichiers de la base de référence et votre fichier de test.

La base de référence et de test

Nous allons créer 3 fichiers :

- Ref_OUI.txt : liste les (chemins des) fichiers OUI de référence (un fichier par ligne)
- Ref_NON.txt : liste les (chemins des) fichiers NON de référence (un fichier par ligne)
- Test.txt : liste les (chemins des) fichiers à tester (un fichier par ligne)
  (=les fichiers que vous avez enregistrés)

Développer l'algorithme DTW

Le programme prendra en entrée les 3 fichiers, et donnera la réponse OUI/NON pour chaque fichier de test.

L'algorithme (pour 1 fichier test comparé à 1 fichier de référence) est le suivant (il est aussi présent dans le cours) :

```
Données
    NB : entier # nbre de coefficients
    n : entier # nbre de trames pour le fichier test
    m : entier # nbre de trames pour le fichier ref
```

```
     X[1..n][NB] : flottant # test
     Y[1..m][NB] : flottant # ref
Variables
     i, j : entier
     M[0..n][0..m] : flottant
Retour
     M[n][m] # le score de l'alignement
DEBUT
     M[0][1..m] = 0
     M[1..n][0] = 0
     Pour i allant de 1 à n
          Pour j allant de 1 à m
               M[i][j] = min(M[i-1][j-1],
                             M[i][j-1],
                             M[i-1][j])+d(X[i],Y[j])
FIN
```

Pour la distance locale `d`, nous prendrons la distance Euclidienne au carré $\Sigma(X|i][k] - Y[j][k])^2 / (\Sigma X[i][k]^2 . \Sigma Y[i][k]^2)$

Le score OUI (par exemple) d'un fichier de test peut être la moyenne des distances par rapport aux OUI de référence, ou encore le minimum de ces distances. On pourra tester les 2 possibilités.

Phase de test

1. Quelle est la distance à OUI et à NON de mots comme NUIT, BUIS, BON, DON ?

2. Comment faire pour rejeter un fichier qui n'est ni OUI, ni NON ?

3. À partir de la sortie de votre programme, calculez le pourcentage de réussite de votre programme en prenant chaque fichier OUI de REF comme test avec les autres fichiers de références (idem avec NON) :

$$\frac{(\boldsymbol{nb\ de\ (bonnes)\ réponses\ OUI} + \boldsymbol{nb\ de\ (bonnes)\ réponses\ NON}}{\boldsymbol{nbre\ de\ fichiers\ de\ test}}$$

On obtient aussi un taux de reconnaissance (éxactitude/*accuracy*) du système (on pourra comparer si l'on prend la distance moyenne ou le minimum de de distance). Donnez aussi les scores pour les OUI seuls et les NON seuls.