TP4 : Reconnaissance de mots isolés par programmation dynamique (DTW)

Le but est de réaliser un système d'assistance au contrôle aérien à travers la reconnaissance de mots isolés dans un contexte de vocabulaire restreint. La méthode de reconnaissance utilisée est fondée sur un alignement temporel dynamique (DTW) ou programmation dynamique.

1. Présentation du système

Des efforts sont réalisés depuis les années 90 afin d'intégrer des systèmes de reconnaissance vocale dans les systèmes de contrôle de trafic aérien, notamment afin de reconnaitre automatiquement les indicatifs aériens des avions en vol.

Le système que vous allez mettre en œuvre sous **Python** se décompose en différentes étapes :

- une lecture du signal audio,
- une extraction des vecteurs de paramètres acoustiques,
- une programmation dynamique (DTW) qui permet de calculer la distance entre le mot prononcé (observation) et un mot du dictionnaire (référence).
- un affichage simple du mot reconnu ou une visualisation détaillée des résultats.

Le dictionnaire est composé des lettres de l'alphabet radio international.

Nous nous limiterons aux treize premières lettres de cet alphabet :
{**alpha**, **bravo**, **charlie**, **delta**, **echo**, **foxtrot**, **golf**, **hotel**, **india**, **juliett**, **kilo**, **lima**, **mike**}.

Une occurrence de chacune des lettres est présente aussi bien en référence (dans le dictionnaire) qu'en observation. Tous les signaux sont fournis au format Wave.

2. Réutilisation de fonctions pour la lecture et la paramétrisation

Réutiliser la fonction « lecture » permettant de lire un fichier son et de connaître ses caractéristiques (fréquence d'échantillonnage, durée).

```
def lecture(fichier, nb_bits):
```

Réutiliser la fonction « parametrisation » qui calcule les paramètres acoustiques (coefficients cepstraux) d'un fichier sonore sur des fenêtres de taille « taille_fenetre » avec un recouvrement de moitié.

```
def parametrisation(signal, taille_fenetre, nbe_coef):
```

Le nombre de coefficients cepstraux « nbe_coef » sera fixé à 8, 12, 16, …, 30.

Modifier votre fonction « parametrisation_total » (utilisant la fonction « parametrisation ») afin qu'elle permette de calculer les paramètres pour l'ensemble des fichiers d'un répertoire « rep_wav » et de stocker les résultats dans un répertoire « rep_mfcc ». Cette fonction renvoie la liste des noms de fichiers traités.
def parametrisation_total(nb_bits, taille_fenetre, nbe_coef, rep_wav, rep_mfcc):

3. Programmation dynamique

Ecrire une fonction « dtw » qui prend deux arguments en entrée : la matrice de coefficients cepstraux du signal à reconnaître (observation) et la matrice de coefficients cepstraux d'un signal de référence. Cette fonction renvoie le coût normalisé.
def dtw(mfcc_ref, mfcc_obs):

Vous utiliserez la distance euclidienne entre chaque couple de vecteurs de paramètres acoustiques. Le développement de cette fonction pourra s'appuyer sur l'algorithme donné en cours/TD.

Ecrire une fonction « dtw_total » (utilisant la fonction « dtw ») qui calcule la DTW sur toutes les observations (fichiers Wave) du répertoire « rep_obs » par rapport à toutes les références (fichiers MFCC) du répertoire « rep_ref ». Cette fonction renvoie une matrice de coûts de taille nb_fichier_obs x nb_fichier_ref.
def DTW_total(nb_bits, taille_fenetre, nbe_coef, rep_ref, rep_obs):

4. Affichages et tests

Affichage simple

Ecrire un programme principal qui lance les fonctions précédentes et affiche pour chaque observation (mot inconnu), le mot le plus probable.

Affichage amélioré

Ajouter à votre programme principal, un affichage des coûts entre une observation et chaque référence sous forme d'histogramme via la commande « bar ».
Calculer le score de reconnaissance.

Tests

Bien évidemment, afin d'améliorer les résultats, vous pourrez modifier :

- le nombre de paramètres (coefficients cepstraux) : nbe_coef,
- la taille de la fenêtre d'analyse : taille_fenetre,

ET TESTER AVEC VOS PROPRES ENREGISTREMENTS !