BE2 – Reconnaissance de la parole

SDI

2021 – 2022

Stéphane Rossignol – stephane.rossignol@centralesupelec.fr

0.1 Extraction des ordres à partir des signaux de parole

0.1.1 La tâche

Votre programme (en Python, ou en Octave/Matlab) devra reconnaître quelques mots/courtes phrases. Nous vous suggérons d'utiliser par exemple : EN AVANT, À DROITE, À GAUCHE, STOP. (L'objectif final, dépassant le cadre de ce BE, par exemple serait de capter les sons (via un micro, une carte-son, etc.), de reconnaître les ordres, et d'envoyer ces ordres à un robot.) Pour commencer, il est nécessaire de tester les différentes étapes et de définir les limites des techniques proposées.

Les outils disponibles sont décrits dans l'annexe .3. Ce sont des indications ; si vous voulez utiliser d'autres outils, faites-le, et indiquez-moi dans votre petit rapport les outils que vous avez utiliss.

0.1.2 Reconnaissance de la parole

Les outils de traitement de la parole se décomposent communément en deux parties : l'extraction de caractéristiques et le décodage. Voir la figure 1 (sur cette figure, nous montrons l'extraction de l'énergie et des MFCC, ainsi que de leurs dérivées première Δ et deuxième Δ^2 : au cours de ce TL, nous nous intéresserons aux MFCC sans leurs dérivées ni l'énergie dans un premier temps, puis vous pourrez ajouter l'énergie et les dérivées des

MFCC et de l'énergie, puis vous pourrez ajouter la LPC (c'est-à-dire, les coefficients autorégressifs : voir le cours).

La première étape consiste à extraire des caractéristiques, via des techniques de traitement de signal appliquées à des trames de son de quelques dizaines de millisecondes. Ces caractéristiques doivent mettre le mieux possible en évidence les particularités des différents segments de la parole. Ces segments peuvent être considérés à des niveaux variés : par exemple, les phonèmes ([a], [e], [i], [t], [s], etc. ; ce qui nous amène à considérer un problème à une trentaine de classes), ou les phonèmes contextualisés ([am], [an], [al], [bl], etc. ; ce qui nous conduit à un problème de plusieurs milliers de classes), ou etc. Les caractéristiques les plus étudiées sont les coefficients de prédiction linéaire (LPC) et les Mèl Filter Cepstrum Coefficients (MFCC). Nous allons aborder en détails les secondes au cours de ce BE.

La deuxième étape se décompose en deux sous-étapes. Premièrement, il s'agit de classifier chacune des trames du son. Comme le nombre de classes est très important, il faut être capable de donner une liste de N meilleures hypothèses. Deuxièmement, ces N meilleures hypothèses sont élaguées, en ajoutant des contraintes telles que les successions possibles de phonèmes, les mots possibles, etc., ce qui permet finalement de reconstruire les mots qui ont/auraient été dits (notez qu'une étape suivante serait d'interpréter ce qui a été dit : nous n'allons pas considérer ceci ici). L'état de l'art indique que le couple GMM (Gaussian Mixture Model) pour le classifieur, et HMM (Hidden Markov Model) pour l'élagage, ou alors les LSTM (deep-learning), est ce qui est communément utilisé. Cependant, ici, la reprogrammation de l'ensemble GMM-HMM ou l'apprentissage de réseaux de neurones profonds n'étant ni envisageables ni nécessaires (la tâche est simple : il n'y a que quatre ordres à distinguer l'un de l'autre), nous allons considérer la DTW (Dynamic Time Warping), qui est une méthode plus aisée à implémenter et qui accomplit les deux sous-étapes simultanément. Cependant, cette méthode ne permettra pas l'utilisation de phrases trop complexes pour les ordres à transmettre (à un robot, par exemple)

0.1.3 Principaux thèmes abordés

— Acquisition des sons
— Caractérisation des sons (dans des conditions stables : mono-locuteur, pas ou peu de bruit... ; puis dans des cas plus généraux : plusieurs locuteurs, ...)
— Décodage par la DTW

0.1.4 Pré-traitement et extraction de caractéristiques

0.1.4.1 Généralités

Comme indiqué, deux types de de caractéristiques vont être étudiées : les coefficients de la prédiction linéaire (voir le cours) et surtout les MFCC (voir l'annexe .1).

Les tests et ajustements des paramètres seront d'abord réalisés sous un logiciel de calcul matriciel (Python ; ou Matlab/Octave pour ceux qui le souhaitent), avec des sons préenregistrés par nous, puis possiblement des sons enregistrés par vous.

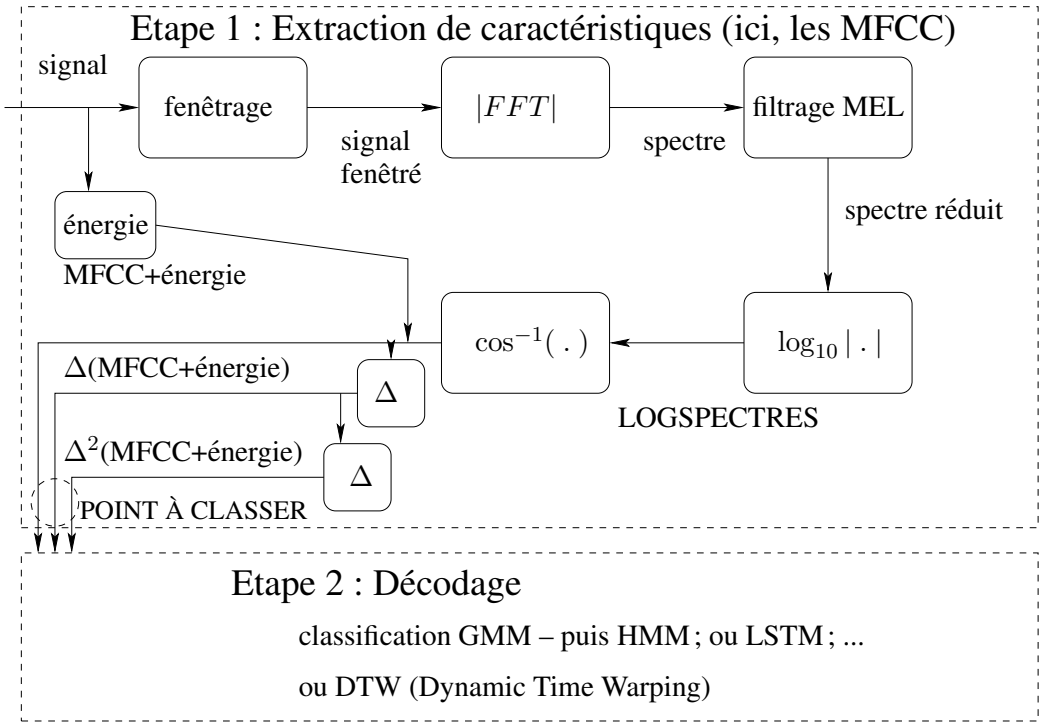

FIGURE 1: Les modules

0.1.4.2 Remarque sur les tests

Nous vous demandons, lors de la validation de cette étape, de contribuer à élargir notre base de données en sauvegardant vos sons et en me les faisant parvenir (attention à la taille de l'archive ! ; une possibilité serait de passer par Teams).

0.1.5 Décodage

Toutes les productions mono-locuteur d'un même son (par exemple, un [a] soutenu), dans les mêmes conditions d'enregistrement, sont différentes, déjà d'un point de vue du contenu spectral. Les caractéristiques permettent normalement (dans le cas idéal) de s'affranchir de ces différences, rendant possible de ce fait une classification efficace. Malheureusement, ces productions diffèrent aussi en longueur ! Par exemple, un locuteur donné ne dit jamais le [o] de STOP exactement avec la même longueur. Pour des mots complets ou des phrases courtes complètes, ce phénomène est encore plus net. La classification n'est donc pas suffisante. Quand on veut comparer deux phrases dites par un locuteur pour décider si la même chose a été dite, il faut les recaler en temps. La DTW est utilisée pour ce faire (voir l'annexe .2).

Pour faire les tests, nous vous suggérons d'utiliser les sons que nous vous fournissons, en prenant comme sons de référence par exemple l'ensemble *agauche2.wav*, *adroite2.wav*, *enavant2.wav* et *stop2.wav*, et comme sons à reconnaître l'ensemble *agauche.wav*, *adroite.wav*, *enavant.wav*, *stop.wav*, *agauche3.wav*, *adroite3.wav*, *enavant3.wav*, et *stop3.wav*.

Il faudra(it) ensuite enregistrer vos propres sons, et tester et régler de nouveau les techniques.

Les tests et ajustements des paramètres seront réalisés sous un logiciel de calcul matriciel (Python ; ou Matlab/Octave), avec des signaux artificiels (s'il le faut), et avec des sons pré-enregistrés par nous, puis possiblement des sons enregistrés par vous.

.1 MFCC

.1.1 Quelques détails sur les MFCC

Le schéma général explicitant le calcul des MFCC (Mel Frequency Cepstral Coefficients) est donné sur la figure 2. En italique, sont données aussi les tailles, parfois en deux dimensions, des signaux extraits.

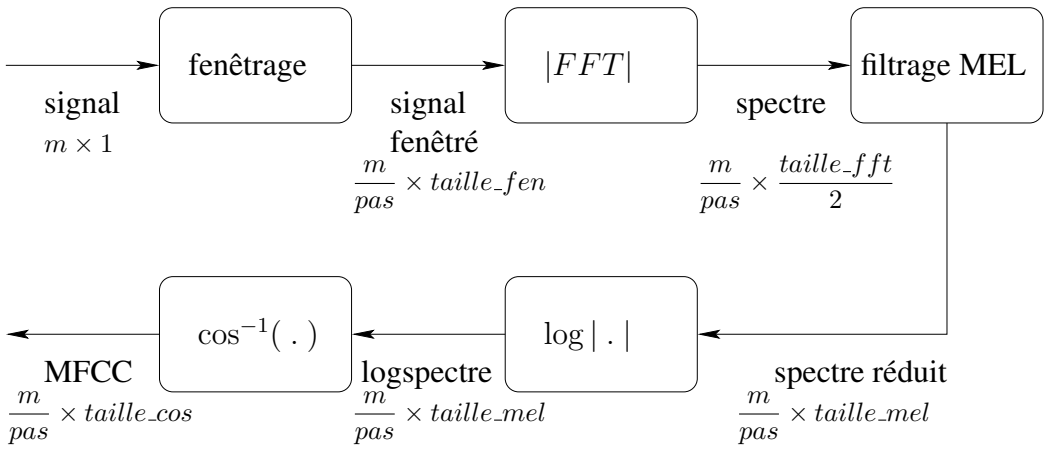

FIGURE 2: Schéma général explicitant le calcul des MFCC

La longueur des fenêtres d'analyse est typiquement de 32 ms. La longueur du pas d'avancement est de l'ordre de 16 ms.

.1.2 Filtres MEL

La formule de conversion F (domaine MEL) vers f (domaine fréquentiel) est :

$$F = 2595 \log_{10}\left(\frac{f}{700} + 1\right)$$

et donc la formule inverse est :

$$f = 700\left(10^{\frac{F}{2595}} - 1\right)$$

Dans le domaine MEL, les filtres sont linéairement espacés, et de largeurs égales. Les bornes d'un filtre donné correspondent aux fréquences centrales des deux filtres qui lui sont adjacents. Dans le domaine fréquentiel, les filtres sont de forme triangulaire. Soit N_f le nombre de filtres considérés, F_{min} la plus petite fréquence MEL considérée et F_{max} la plus grande. La largeur de chaque filtre dans le domaine MEL est :

$$W = 2\frac{F_{max} - F_{min}}{N_f + 1}$$

Alors, les trois fréquences importantes, c'est-à-dire celles de chacune des deux bornes et la fréquence centrale, sont respectivement égales à, pour le filtre de numéro d'ordre i (avec $i = 1 \rightarrow N_f$) :

$$F_{center}^{(i)} = F_{min} + i\frac{W}{2},$$

$$F_{min}^{(i)} = F_{min} + (i-1)\frac{W}{2}, \text{ et}$$

$$F_{max}^{(i)} = F_{min} + (i-1)\frac{W}{2} + W$$

et il est aisé de trouver les $f_{min}^{(i)}$, $f_{centre}^{(i)}$ et $f_{max}^{(i)}$ correspondant. Comme les filtres sont triangulaires dans le domaine fréquentiel, et de réponse unitaire pour $f_{centre}^{(i)}$, la réponse en fréquence pendant la phase ascendante a et la réponse en fréquence pendant la phase descendante b sont aisées à calculer :

$$a = \frac{f - f_{min}^{(i)}}{f_{centre}^{(i)} - f_{min}^{(i)}} \text{ pour } f \in [f_{min}^{(i)} \;\; f_{centre}^{(i)}]$$

et

$$b = \frac{f_{max}^{(i)} - f}{f_{max}^{(i)} - f_{centre}^{(i)}} \text{ pour } f \in [f_{centre}^{(i)} \;\; f_{max}^{(i)}]$$

.1.3 La FFT

Il ne vous est pas demandé de programmer la FFT. Il existe des bibliothèques toutes prêtes, dans tous les langages : voir FFT, sous Python, Matlab, Octave, Julia, etc.

.1.4 La DCT

$\cos^{-1}$ est une notation correspondant à la transformation en cosinus inverse (iDCT) d'un signal échantillonné X. Il existe un certain nombre de définitions pour la transformation en cosinus inverse. Celle qui peut être utilisée ici est :

$$x[n] = \sum_{k=0}^{N-1} w[k]X[k]\cos\left(\pi\frac{(2n-1)k}{2N}\right), \quad n = 0,\ ...,\ N-1$$

avec : $w(0) = \sqrt{\frac{1}{N}}$ et $w(k) = \sqrt{\frac{2}{N}}, \quad k = 1, ..., N-1$

De plus, certains auteurs utilisent directement la DCT au lieu de la iDCT.

.2 DTW

Il s'agit dans un premier temps de déterminer k sons de référence. Ici, typiquement, $k = 4$, puisqu'il y a quatre ordres différents à reconnaître. Mais ceci peut-être raffiné (plusieurs sons par ordre si plusieurs locuteurs, etc.) ! Comme indiqué dans l'énoncé, vous pouvez commencer par prendre *agauche2.wav*, *adroite2.wav*, *enavant2.wav* et *stop2.wav*.

Il faut calculer une distance entre le son à reconnaître et chacun des sons de référence, et la distance la plus petite nous donne la solution.

Le problème est de définir une « distance » entre deux sons.

.2.1 Distance locale

D'ores et déjà, nous ne nous travaillons pas directement avec les sons, mais avec les caractéristiques extraites. Supposons que le son à reconnaître ait donné lieu à l'extraction de N trames et que le son de référence k ait donné lieu à l'extraction de J^k trames. Nous formons la matrice D des distances locales $d(x_n, y_j^k)$ qui est de dimension $N \times J^k$. x_n correspond au vecteur des caractéristiques extraites pour la trame numéro n du son à reconnaître et y_j^k correspond au vecteur des caractéristiques extraites pour la trame numéro j du son de référence k. La distance considérée peut être par exemple la distance euclidienne :

$$d(x_n, y_j^k) = \sqrt{\sum_{i=1}^{I} (x_{ni} - y_{ji}^k)^2}$$

où I est le nombre de caractéristiques extraites.

D'autres distances locales peuvent être utilisées.

.2.2 Meilleure distance globale

Pour rechercher la meilleure distance globale entre la séquence à reconnaître et la séquence de référence, il suffit alors de rechercher le chemin optimal dans la matrice D de façon à minimiser la somme des distances locales rencontrées pour aller d'un point initial (généralement $(1, 1)$, correspondant au début des deux séquences) à un point final (généralement $(N, J(k))$, correspondant à la fin des deux séquences).

Il peut être montré que la distance optimale est obtenue en calculant, pour chaque entrée (n, j) la distance cumulée $D(n, j)$ correspondant à la distance optimale qui est obtenue en comparant les deux sous-séquences correspondant aux n premiers vecteurs à reconnaître et aux j premiers vecteurs de référence. Il peut être montré que cette distance peut se calculer selon la récurrence suivante :

$$D(n, j) = d(n, j) + \min_{p(n,j)} \{D(p(n, j))\}$$

où $p(n, j)$ représente l'ensemble des prédécesseurs possibles de (n, j), définis de façon à obtenir une trajectoire monotone et plausible.

Dans le cas le plus simple, et en général, nous prenons :

$$p(n,j) = \{(n-1,j)\ (n,j-1)\ (n-1,j-1)\}$$

Il y a d'autres contraintes possibles.

Sur la figure 3, nous présentons la contrainte donnée ci-dessus et un exemple de chemin obtenu avec $I = 1$ et les caractéristiques extraites $[4231]$ et $[44422231]$ pour les deux séquences à comparer (les valeurs des caractéristiques sont artificiellement reportées sur les axes).

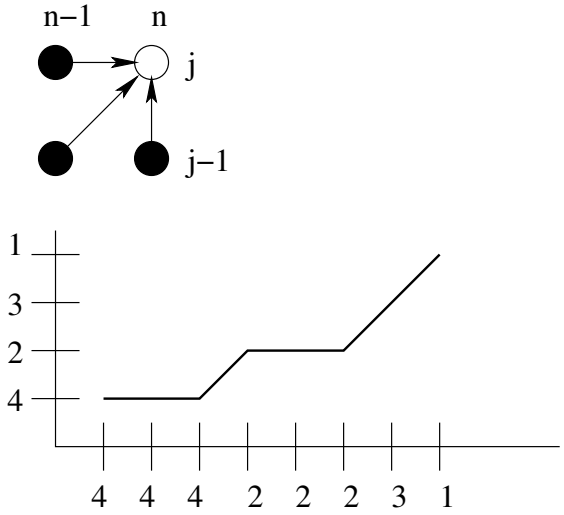

FIGURE 3: La DTW

Un dernier traitement peut être effectué, qui consiste à normaliser la distance D par le nombre de trames du signal de référence. Ceci permet de comparer des mots de longueurs très différentes (comme « stop » et « en avant »). D'autres méthodes de normalisation peuvent être envisagées.

.3 Outils disponibles

.3.1 Implantations de l'extraction des MFCCs

— MFCC en Python : `https://python-speech-features.readthedocs.io/en/latest/#`.
— MFCC en Octave : `https://github.com/jagdish7908/mfcc-octave` (permet aussi d'enregistrer les sons ; mais peut aussi traiter des sons pr-enregistrs : voir lignes 9 à 11 de "mfcc.m").
— MFCC en Matlab : c'est dans l'"Audio toolbox".

.3.2 Implantations du calcul de la DTW

— Liens vers un certains nombres de langages de programmation, dont Python (plusieurs possibilités) : `https://en.wikipedia.org/wiki/Dynamic_time_warping`.
— DTW en Octave : voir `https://stackoverflow.com/questions/36675738/dtw-algorithm-oct-file`.
— DTW en Matlab : c'est dans la "Signal processing toolbox".