UMONS
Université de Mons

Faculté des Sciences

Traitement du signal : Projet

Speaker classification

I-ISIA-030

Réalisé par Louis DASCOTTE
& Nicolas DELPLANQUE
& Nicolas SOURNAC

Faculté
des Sciences

3e Bachelier en Sciences Informatiques
Année 2021-2022

ACADÉMIE
UNIVERSITAIRE
WALLONIE-
BRUXELLES

Résumé

Ce rapport contient l'ensemble des résultats obtenus, leurs interprétations ainsi que les explications du fonctionnement de notre implémentation du projet de traitement du signal. Ce projet consiste en la réalisation de solutions visant à classifier des personnes en fonction de leur genre à partir d'enregistrement de leur voix.

Table des matières

1. Exécution du code

1.1 Librairies utilisées

1.2 Structure du code source

2. Caractéristiques étudiées

Pour la réalisation du projet, nous avons été amenés à étudier différentes caractéristiques des signaux de parole en vue de les utiliser afin de pouvoir classifier les différents speakers. Pour chacune de ces caractéristiques, nous avons donc implémenté un algorithme qui permet de la calculer et nous avons ensuite analysé et interprété les résultats afin d'en déduire des règles sur lesquelles la classification allait se baser.

2.1 Energie du signal

L'énergie du signal est calculée à partir de la fonction compute_energy() qui prend un signal en paramètre. Cette fonctionne parcourt simplement l'entièreté du signal en question et y applique la formule du calcul de l'énergie. Nous utilisons les résultats du calcule de l'énergie du signal afin de déterminer si une frame est voiced ou unvoiced.

2.2 Fréquence fondamentale

Le pitch ou la fréquence fondamentale représente la plus basse fréquence qui constitue un signal. Cette valeur doit varier entre 60 Hz pour les voix les plus graves et 550 Hz pour les voix les plus aigües.
Lors du projet, nous avons été amenés à calculer la fréquence fondamentale des speakers en utilisant deux méthodes différentes : La méthode basée sur l'autocorrelation et la méthode basée sur les cepstrums. Pour ce faire, nous avons créé les fonctions autocorrelation_pitch_estim() et cepstrum_pitch_estim().
Pour la méthode basée sur l'autocorrelation, l'algorithme doit recevoir en paramètre une liste de minimum 5 fichiers audio. Nous lisons chaque fichier, normalisons son signal, le divisons en frames de 50 ms et récupérons uniquement les frames dont l'énergie est supérieure au seuil que nous avons déterminé graphiquement. Ensuite, pour chacune des frames restantes, nous calculons sa corrélations avec un maxlags qui vaut 200. Grâce à la fonction find_peaks() fournie par la librairie numpy, nous récupérons facilement l'ensemble des pics du signal corrélé. Il ne nous reste plus qu'à calculer la distance en Hz entre les deux plus hauts pics pour obtenir la fréquence fondamentale de la frame. La fonction autocorrelation_pitch_estim() retourne la moyenne de toutes les fréquences fondamentales calculées à partir des frames en retirant les fréquences supérieures à 550 Hz considérées comme erronées. Nous considérons ce résultat étant comme la fréquence fondamentale de la personne à l'origine des fichiers audio.

Pour la méthode basée sur les cepstrums, le début de l'algorithme est similaire au précédent. Nous lisons également chaque fichiers audio fournis, les normalisons, les divisons en frames de 50 ms et récupérons uniquement les frames dont l'énergie est supérieur au seuil que nous avons déterminé graphiquement. Ensuite, pour chacune des frames restantes, nous calculons dans un premier temps son cepstrum de manière brut et dans un deuxième temps son cepstrum après lui avoir appliqué une fenêtre de hamming. Il suffit ensuite de trouver l'indice du plus haut pics présents dans ces deux cepstrums dans l'interval allant de 60Hz à 550Hz et de le convertir en Hz pour obtenir la fréquence fondamentale de la frame. La fonction cepstrum_pitch_estim() retourne la moyenne de toutes les fréquences fondamentales calculées à partir des frames. Nous considérons ce résultat comme étant la fréquence fondamentale de la personne à l'origine des fichiers audio.

Afin de déterminer des règles de différenciations sur base de la fréquence fondamentale, nous avons exécuté 15 fois les deux algorithmes en donnant des fichiers des deux speakers. Pour BDL nous avons pris la valeur maximum obtenues sur les 15 itérations et l'avons considérées comme borne supérieure et pour SLT nous avons pris la valeur minimum obtenues sur les 15 itérations et l'avons considérées comme borne inférieure. Cela nous a donné les règles suivantes :
Pour la méthode basée sur l'autocorrelation, si f0 $<$ 145Hz alors c'est BDL et si f0 $>$ 170Hz alors c'est SLT.
Pour la méthode basée sur les ceptrums, si f0 $<$ 166Hz alors c'est BDL et si f0 $>$ 217Hz, alors c'est SLT.

2.3 Formants

Les formants sont les pics observés sur l'enveloppe de la réponse en fréquence. Dans le cadre de ce projet, l'estimation des formants se base sur un algorithme LPC. Pour ce faire, nous avons créé la fonction compute_formant() qui prend un paramètre un fichier audio. L'algorithme va lire le fichier, le normaliser, et le diviser en frames de 25 ms. Pour chacune des frames, un filtre passe-haut du premier ordre lui est appliqué suivi d'une fenêtre de hamming. Nous calculons ensuite les coefficients LPC et trouvons ensuite les racines associées. Les formants sont directement déduits de ces racines complexes. L'algorithmes retourne donc une liste de formants correspondants aux formants de chacune des frames du fichier audio.

2.4 MFCCs

3. Systèmes basés sur des règles

Bien expliquer les différents systèmes + comment on les utilise + comment on a fait pour estimer leur précisions + quelles données on a utiliser pour les tester

3.1 Système 01

3.2 Système 02

3.3 Système 03

4. Machine learning