Université de Technologie de Troyes

Systèmes, Réseaux et Télécommunications
Technologies Mobiles et Systèmes Embarqués

Reconnaissance du locuteur avec SVM

Victor Martin
<victormartinfr@gmail.com>

Troyes, 29 janvier 2014

Table des matières

Chapitre

1 Introduction

Dans le cadre d'une TX encadrée par Hichem SNOUSSI, j'ai du réaliser une application de reconnaissance vocale, et plus précisement une application de reconnaissance du locuteur. Le but est que l'application puisse analyser des fichiers audios de personnes qui parlent, en extraire des caractéristiques propres à l'élocution de chaque personne et indépendantes du texte qui est prononcé, et pouvoir créer un classifieur grâce à des SVMs. Enfin, on peut lui fournir en entrée un fichier audio d'une personne dont la voix est dans la base d'apprentissage, et le programme est censé être capable d'identifier la personne locutrice.

Dans ce rapport, je présenterais les techniques pour l'extraction de caractéristiques et pour l'apprentissage, en insistant sur celles que j'ai utilisé. Ensuite, j'exposerais les différentes fonctionnalités et bibliothèques utilisées par mon application.

Chapitre 2

Extraction de caractéristiques dans la voix

2.1 Présentation des différentes possibilités

Pour pouvoir faire de la reconnaissance d'un locuteur, il faut pouvoir extraire certaines caractéristiques [1] de sa voix. Ces caractéristiques sont divisées en cinq niveaux [1] :

2.1.1 Caractéristiques spectrales à court-terme

Les caractéristiques spectrales à court-terme comme le timbre de la voix, qui sont les particularités de plus bas niveaux. Ici, nous n'avons besoin que de courtes séquences ($\leqslant$ 30ms) sonores pour pouvoir identifier une voix, ce qui permet une reconnaissance en temps-réel de manière indépendante du texte prononcé. Par contre, les erreurs peuvent être plus facile à faire et la qualité de l'identification est plus sensible au bruit.

2.1.2 Caractéristiques de la source la voix

Les caractéristiques de la source la voix comme la fréquence fondamentale ou encore le taux d'ouverture des cordes vocales.

2.1.3 Caractéristiques spectro-temporelles

Les caractéristiques spectro-temporelles de la voix (modulation en amplitude, modulation en fréquence, temps-fréquence en composantes principales, filtres temporels dirigés par les données) qui peuvent être extraites entre autres par des transformées en cosinus discretes sur des segments audio de $\sim$300ms.

1. On retouve surtout le terme de feature extraction dans la littérature

2.1.4 Caractéristiques prosodiques

Les caractéristiques prosodiques [2] comme l'intonation, le rythme ou encore la modulation de la prononciation des syllabes influencée par les émotions du locuteur.

2.1.5 Caractéristiques de plus haut niveaux

Enfin, les caractéristiques de plus haut niveaux comme l'usage de certains mots, les "tics de langage" ... Dans ce cas, il faut réaliser une véritable reconnaissance automatique du langage naturel, ce qui demande une large base d'apprentissage, plus d'échantillons à identifier (> 1s) et rallonge le temps de calcul et donc retarde la prise de décision. À l'inverse, ce traitement est beaucoup moins sensible au bruit que le traitement bas niveaux.

2.1.6 Conclusion

Ces cinq catégories allant des plus bas niveaux aux plus haut niveaux présentent des avantages et des inconvénients. Les caractéristiques de plus bas niveaux demandent des échantillons assez petits et permettent ainsi de faire de la reconnaissance en temps-réel mais sont plus sensibles au bruit, et inversement pour les caractéristiques de plus haut niveaux.
On peut voir une représentation de ces différents niveaux sur la figure 2.1.

2. Prosodie : "Étude de la forme et de la substance des éléments phoniques dont les limites ne coïncident pas avec celles du phonème, qu'elles soient inférieures (comme les mores) ou supérieures (comme la syllabe, le mot, le syntagme et la phrase). Ainsi, les recherches sur l'accent et sur l'intonation ; l'étude du rythme, du débit et des pauses dans la parole." - www.larousse.fr

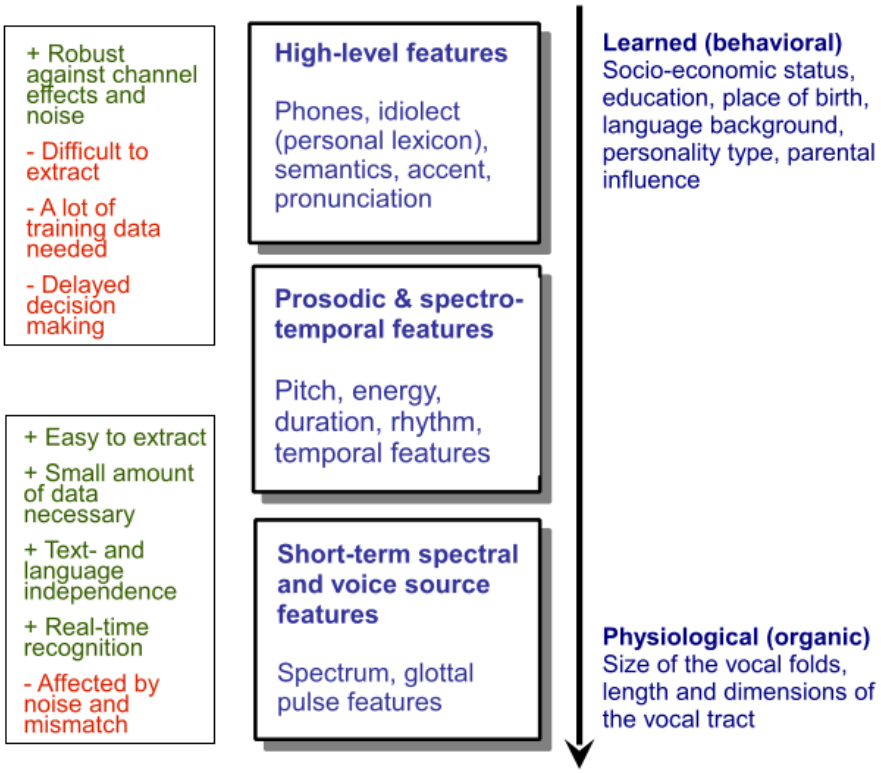

FIGURE 2.1 – Une illustration de [1] des différentes caractéristiques sur plusieurs niveaux d'abstraction

Voulant effectuer un traitement rapide du son pour mon application et ne necessitant pas une grande base d'apprentissage, mon choix s'est porté sur les caractéristiques spectrales à court-terme de la voix.

2.2 Choix effectué

J'ai choisi d'étudier les caractéristiques spectrales à court-terme de la voix, et plus précisement, le cepstre [3]. Le cepstre est la tranformée de Fourier du signal, suivie d'un logarithme, et enfin suivie par une transformée de Fourier inverse. Il permet de donner des informations sur le taux de variation sur plusieurs bandes spectrales.
L'analyse de ce cepstre permet d'obtenir des informations sur le timbre de la voix, et notamment grâce à l'extraction des MFCCs [4]. Les MFCCs sont des coefficients permettant de caractériser numériquement le timbre à partir du

3. Note : cepstre est l'inverse du mot spectre pour désigner qu'on part du domaine fréquentielle vers le domaine temporelle, ce qui est l'inverse pour obtenir un spectre.

4. Mel-frequency cepstral coefficients

cepstre.
Pour obtenir ces coefficients, l'algorithme à suivre est le suivant :

1. Découper le signal audio en plusieurs segments (ou fenêtres) dans le domaine temporel (fenêtre entre 20ms et 30ms).
2. Appliquer un fenêtrage de Hamming sur chaque segment.
3. Appliquer une transformée de Fourier sur chaque segment
4. Transposer chaque spectre sur l'échelle de Mel [5] :

$$mel(f) = 1127 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{2.1}$$

   où f est la fréquence à transposer dans l'échelle de Mel.
5. Prendre le log de la puissance de chaque fréquence en Mel.
6. Prendre la transformation en cosinus discrète de ces logarithmes, comme si c'était un signal.
7. Retenir les 13 coefficients les plus bas du résultat de cette transformation

Ainsi, nous avons pour chaque fenêtre de 20-30ms, un vecteur de dimension 13 caracterisant le timbre de la voix sur ce segment. Il reste ainsi à créer le méchanisme d'apprentissage pour utiliser tous ces vecteurs.

5. Échelle de Mel : "Elle a été conçue de telle façon que 1 000 Hz correspondent à 1 000 mels et qu'une variation constante dans le domaine des mels soit perçue comme une variation constante de fréquence (en Hz) par les auditeurs. En effet, au-delà de 500 Hz, l'oreille humaine ne perçoit plus le changement d'octave comme un doublement de la fréquence. Par contre, si on se place sur une échelle de Mel, un changement d'octave sera effectivement perçu comme un doublement de la fréquence." - Wikipédia
Plus d'informations sur l'échelle de Mel : [3]

Chapitre

3 Apprentissage

Dans ce chapitre, nous allons étudier certaines techniques d'apprentissage supervisé.

3.1 Gaussian Mixture Model

Les modèles à mélanges de gaussiennes modélisent un ensemble de données expérimentales comme étant distribué selon une somme de lois gaussiennes dont il faut trouver les paramètres (moyenne, variance, amplitude) de chacunes des lois. Ces différents paramètres sont alors estimés grâce à un algorithme d'espérance-maximisation (algorithme EM).
On peut voir une illustration de cette technique sur la Figure 3.1.

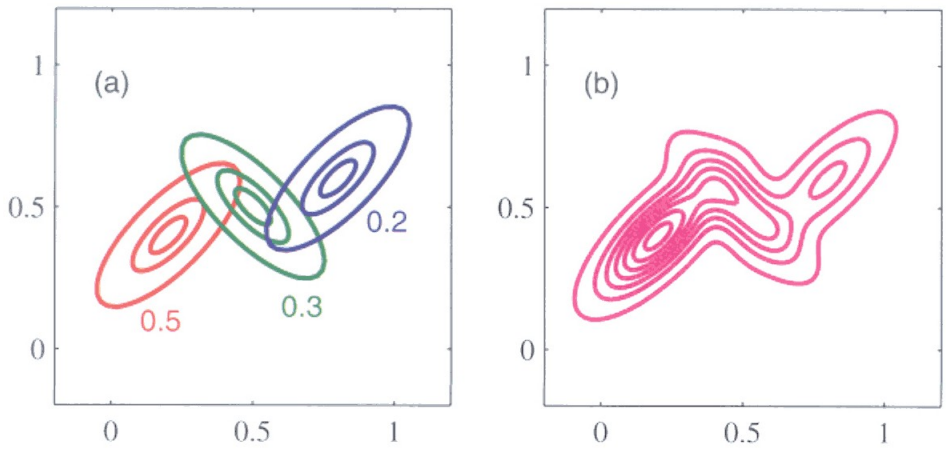

Figure 3.1 – Un exemple où un ensemble de données va être assimilable à la distrubtion de trois gaussiennes de paramètres différents. Sur le schéma (b), les trois gaussiennes séparées ont été additionnées : c'est un mélange de gaussiennes

TABLE I
GMM IDENTIFICATION PERFORMANCE FOR DIFFERENT AMOUNTS OF TRAINING DATA AND MODEL ORDERS

Amount of Training Speech	Model Order	Test Length		
		1 sec	5 sec	10 sec
30 sec	$M = 8$	54.6	79.8	85.6
	$M = 16$	63.7	87.3	90.5
	$M = 32$	64.6	85.3	88.4
60 sec	$M = 8$	66.1	91.5	97.3
	$M = 16$	74.9	95.7	98.8
	$M = 32$	78.6	95.6	98.3
90 sec	$M = 8$	71.5	95.5	98.8
	$M = 16$	79.0	98.0	99.7
	$M = 32$	84.7	98.8	99.6

FIGURE 3.2 – Résultat d'une étude [5] sur l'utilisation des GMMs pour de l'identification de locuteur. Pourcentage de réussite en fonction de la durée des fichiers de la base d'apprentissage, du fichier à identifier et du nombre de gaussiennes.

3.2 Réseaux de neurones

Également, les réseaux de neurones sont utilisés pour réaliser de la reconnaissance de locuteur. Un neurone sera modélisé informatiquement comme une entité prenant plusieurs entrées, realisant une combinaison de ces entrées (les coefficients de cette combinaison sont a calculer grâce à la base d'apprentissage, ce sont les poids sur la Figure 3.3) et renverra une sortie dont la valeur dependra de si la combinaison atteint ou non un certain seuil. Ainsi, un réseau de neurones sera un ensemble de neurones où certains d'entre eux auront leurs sorties connecter à une entrée d'un autre neurone du réseau.
Les réseaux de neurones ont déjà été utilisés avec succès pour effectuer de la reconnaissance de locuteur [4].

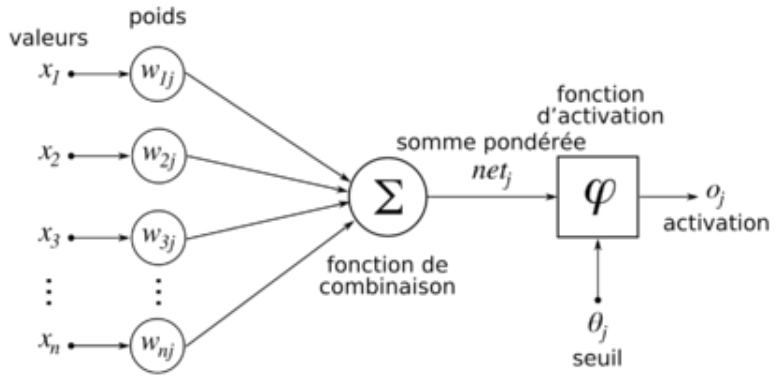

FIGURE 3.3 – Ici, le fonctionnement d'un neurone informatique.

3.3 Support Vector Machines

Les Machines à Vecteurs de Support (SVM) sont des types de classifieurs assez puissants, et qui ont une très bonne faculté de généralisation, surtout comparée à des réseaux de neurones artificiels précedemment évoqués. Le but des SVMs va être de chercher un hyperplan qui va séparer linéairement deux classes de données, tout en maximisant la marge entre entre l'hyperplan séparateur et les données (son fonctionnement est illustré sur la Figure 3.4) .
Si deux classes de données ne sont pas linéairement séparables, on peut transformer l'espace des données (grâce à une transformation Φ) en un espace de dimension supérieure, qui peut être infinie, où les données sont séparables par un hyperplan. En pratique, ce nouvel espace de dimension potentiellement infinie induit des difficultés pour travailler avec ces vecteurs transformés. Cependant, grâce au coup du noyau ("kernel trick"), on peut utiliser un noyau de Mercer k pour connaitre le résultat du produit scalaire de deux vecteurs de l'espace transformé, sans connaitre la valeur des vecteurs transformés ni même la transformation Φ.

$$k(x_i, x_j) = \Phi(x_i).\Phi(x_j) \tag{3.1}$$

où x_i et x_j sont des données de l'espace des données et $\Phi(x_i)$ et $\Phi(x_j)$ sont les données dans le nouvel espace de dimension supérieure.

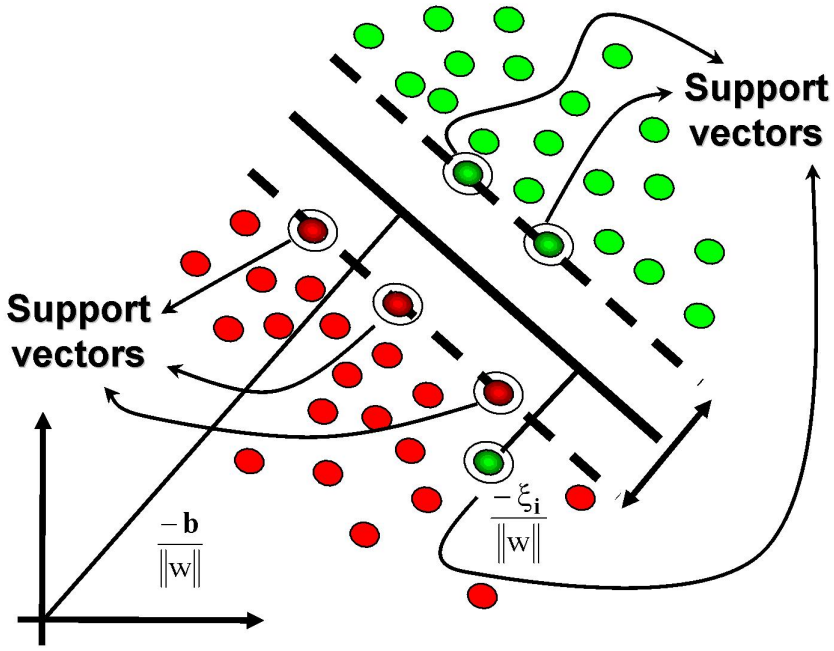

FIGURE 3.4 – L'application d'une SVM sur deux classes (classe verte et classe bleue) de vecteurs de dimension deux. Les vecteurs de support mis en évidence dans chaque classe servent de référence pour la maximisation de la marge entre la séparatrice et les classes. Une fois ce classifieur trouvé (la droite séparatrice), pour connaitre la classe d'appartenance d'un point non étiqueté, il suffit de savoir de quelle côté de la droite est ce point.

Pour mon application, c'est les SVMs que j'ai choisi pour effectuer de la reconnaissance de locuteur.

Chapitre 4

Présentation de mon application

4.1 Comportement fonctionnel

4.1.1 Explications

Mon application va lire n fichiers audio, extraire 13 MFCCs toutes les ~20ms pour chaque fichier [1], puis entrainer une SVM avec ces données étiquetées de 1 à n (voir la Figure 4.1 pour une illustration). Ensuite, l'application va ouvrir un autre fichier audio de voix humaine à identifier, extraire des MFCCs de la même manière, et pour chaque vecteur, essayer de le classifier avec la SVM entrainée (voir la Figure 4.2 pour une illustration).
Ainsi, le nombre de vecteurs de l'extrait audio à identifier étiquetés à la classe $i \in 1 \dots n$ divisé par le nombre de vecteurs totaux, donnera la probabilité que cet extrait à été prononcé par la personne i.
Le type de machine SVM utilisée est une machine C-SVC, pour faire la classification multi-classes.

4.1.2 Schémas fonctionnels

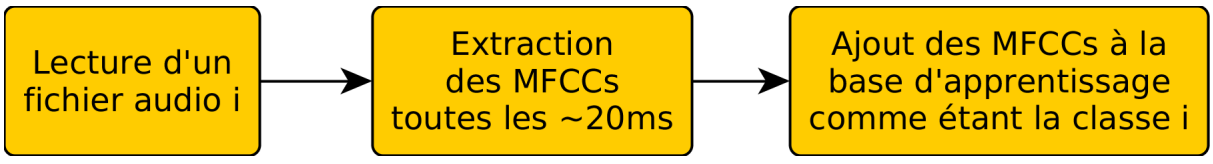

Figure 4.1 – La chaîne de traitement pour l'ajout d'un fichier à la base d'apprentissage.

1. Pour donner un ordre d'idée : pour un son de 10 secondes, l'application extraira approximativement $\frac{10}{0.02} = 500$ vecteurs de dimension 13.

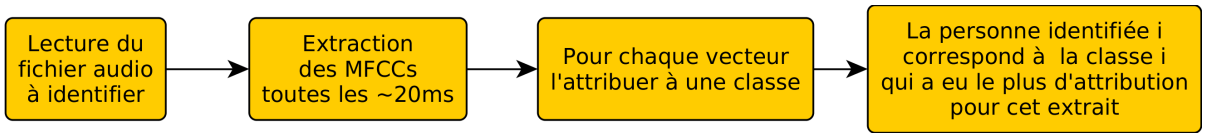

FIGURE 4.2 – La chaîne de traitement pour l'identification d'une voix humaine.

4.2 Implémentation

L'implémentation du logiciel est réalisée en langage C. Il utilise les bibliothèques suivantes :

- libsvm qui permet de faire de la classification (C-SVC, nu-SVR), de la regression (epsilon-SVR, nu-SVR) et de l'estimation de distribution (one-class SVM). Elle supporte la classification multi-classes.
- aubio qui permet de lire des fichiers audio WAVs et qui réalise une extraction de MFCCs.
- IUP qui permet de générer des interfaces graphiques. C'est une bibliothèque très peu connue développée par des ingénieurs de l'université PUC-Rio (Brésil) qui permet de concevoir rapidement des interfaces utilisateurs sur Windows, Mac OS et Linux en utilisant les fonctions natives du système d'exploitation hôte.

4.3 Capture d'écran

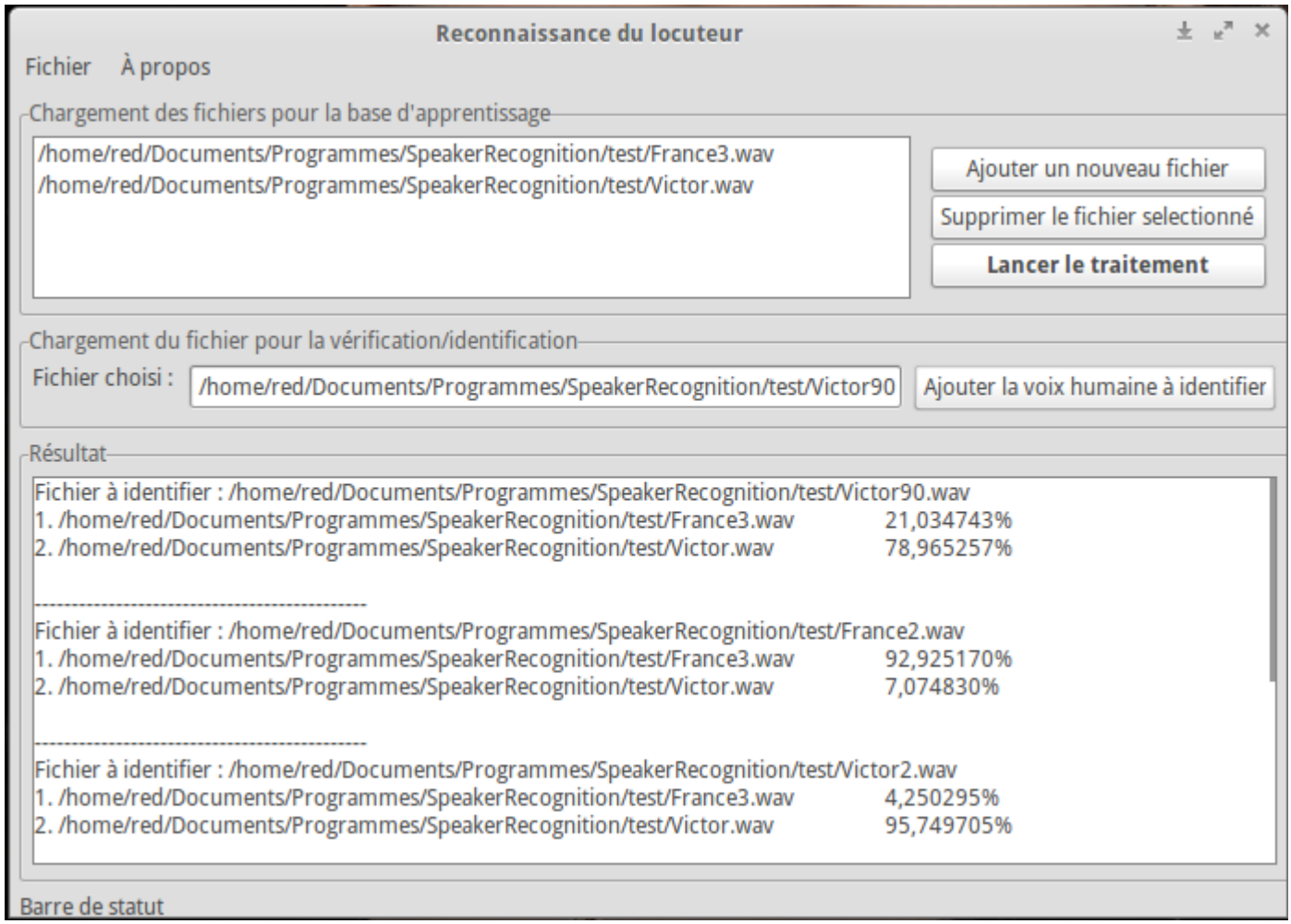

FIGURE 4.3 – Capture d'écran de l'application. La première zone sert à charger la base d'apprentissage. La deuxième zone permet d'ajouter le fichier à identifier. Et enfin, la troizième zone affiche le résultat.
On voit ici que le fichier Victor2.wav est attribué à la voix du fichier Victor.wav à ∼96%.

4.4 Résultats & Limites

Pour effectuer des test, j'ai utilisé ces paramètres :
- Algorithme : C-SVC
- Type du noyau : RBK (Gaussien)
- Gamma : 1/13 (car 13 MFCCs)

– C : 1

Avec trois extraits de ∼12 secondes de deux personnes distinctes, ma base d'apprentissage et d'identification est bien trop petite pour avoir des résultats représentatifs.
Cependant, on peut noter que les résultats que j'ai obtenu sont très positifs : dans la très grande majorité des cas, les voix à identifier ont été correctement identifiées avec un pourcentage de confiance de plus de 90%.
J'ai remarqué que la qualité des résultats était sensible au bruit et aux différences de volumes sonores :

- un extrait enregistré sur un micro defectueux comportant un son aigu en parallèle n'est pas bien identifié
- un autre extrait que j'ai enregistré en parlant très près du micro n'est pas correctement identifié. Bien que le son sature dans cet extrait, ma voix reste reconnaissable par un être humain. Il y a donc ici une piste d'amélioration.

Chapitre 5

Conclusion

En conclusion, on peut remarquer que l'application que j'ai réalisé obtient des résultats prometteurs, bien que le nombre insuffisant de tests ne permettent pas de dire que l'identification fonctionne tout le temps.
On peut envisager certaines pistes d'améliorations :

- Optimiser les paramètres par méthode de cross-validation : changer un peu les paramètres et relancer une identification et voir si ce changement a amélioré les résultats, pour ainsi converger vers un optimum.
- Les concepteurs de la bibliothèque LibSVM soulèvent dans un article l'importance de la normalisation des données. En prenant des données issues de l'industrie, ils montrent une très nette amélioration de la classification après avoir mis toutes les données (base d'apprentissage + données à identifier) dans le même intervalle $[0\,;\,1]$, réduisant ainsi les erreurs dû aux instabilités numériques.
- Également, on peut essayer d'utiliser une one class SVM pour effectuer de la vérification. C'est ce que j'ai voulu faire en premier lieu mais j'ai eu des résultats incohérents : en essayant de classifier les mêmes données de la base d'apprentissage, seul 50% des données étaient correctement classifiées, au lieu de 90% à 100%.
- On peut aussi améliorer la qualité de la reconnaissance en utilisant d'autres features, et notamment les LPCC (Linear Predictive Coding Coefficients) qui sont des coefficients bas niveau très utilisés pour compresser de la voix humaine.
- Avoir un plus grande base d'apprentissage pour faire de vrais tests et pouvoir faire de la reconnaissance du sexe à partir de la voix : en mettant plusieurs voix d'hommes dans une classe et plusieurs voix de femmes dans une autre, on peut facilment avoir un classifieur qui peut reconnaitre le genre du locuteur.

Chapitre 5

Bibliographie

[1] Kinnunen T., Lib H., *An Overview of Text-Independent Speaker Recognition : from Features to Supervectors.* July 2009.

[2] Gudnason J., Brookes M. *Voice source cepstrum coefficients for speaker identification.* 2008.

[3] Combrinck H.P., Botha E.C. *On The Mel-scaled Cepstrum.*

[4] Marhon S., Ubaid Al-Aghar D. *Speaker Recognition Based On Neural Networks.*

[5] Reynolds D., Rose R. *Robust text-independent speaker identification using Gaussian Mixture Speaker models.*