



SPEECHAPP : un logiciel de reconnaissance automatique de la parole

[projet du MIG *Systèmes Embarqués* 2013]

Julien CAILLARD, Adrien DE LA VAISSIÈRE, Thomas DEBARRE,
Matthieu DENOUX, Maxime ERNOULT, Axel GOERING,
Clément JOUDET, Nathanaël KASRIEL, Anis KHLIF,
Sofiane MAHIOU, Paul MUSTIÈRE, Clément ROIG, David VITOUX

18/11/13 - 6/12/13



Remerciements

Toute l'équipe SE souhaiterait tout d'abord remercier **Valérie Roy** pour son engagement, ses bons conseils et son dynamisme. Nous remercions aussi tout particulièrement **Catherine Auguet-Chadaj** pour son accueil chaleureux au Centre de Mathématiques Appliquées lors des trois semaines de stages ainsi que le jury pour ses précieux conseils lors de notre soutenance orale.

Dans le cadre de ces trois semaines, nous avons effectué plusieurs visites et nous voulons pour cela remercier les ingénieurs, cadres, directeurs des ressources humaines, mécaniciens, commerciaux et toutes les autres personnes qui nous ont accueillis, dans l'ordre chronologique :

- Sur la rame **IRIS**, merci à Guillaume FOUILLET de la direction technique IG-T de la SNCF
- À **Eurocopter** et **HELISIM**, un grand merci à Kodor ABOU-TAHAR (EPRE) et Coralie CHOUMAN (EERP) qui nous ont suivis tout au long de la journée. Merci aussi à Vincent VELCKER (ETZWWF) qui nous a très clairement expliqué les systèmes avioniques et la vérification qui accompagne la réalisation des logiciels embarqués, Marc Saisset (EDDDA) pour ses explications sur les moteurs d'hélicoptères, Nicolas JAUNARD qui forme les pilotes d'hélicoptères et nous a permis de monter dans la cabine d'un de ces bijoux de technologie, Ronan PITOIS et enfin Géraud PARJADIS (ETS), employé d'HELISIM pour son exposé sur la compagnie et pour les quelques minutes passées dans un simulateur
- À **Dassault-Aviation**, nous remercions Nicole MARY et Roland MIGINIAC pour leur accueil et leurs nombreuses explications sur la société. Nous remercions aussi tous les intervenants que nous avons pu croiser pendant la journée et qui nous ont permis de découvrir de plus près les avions fabriqués par Dassault ainsi que les simulateurs utilisés par l'entreprise afin de mettre au point leurs systèmes embarqués
- À **Esterel Technologies**, merci à Gunther SIEGEL (VP Engineering) qui nous a présenté les activités de la compagnie et de la R&D ainsi que le principe de certification de leurs logiciels, merci à Frederic BESSIERE (VP ESEG) qui nous a fait une démonstration de leurs produits (Scade Suite/System/Display couplés aux logiciels Ansys), à Gerard MORIN (VP professional services) qui nous a exposé les impacts méthodologiques de l'adoption des produits de la suite Scade chez nos clients et enfin à Amar BOUALI (VP sales southern europe) sur les aspects commerciaux (et son parcours original du laboratoire de recherche jusqu'à la fonction commerciale)

Nous avons aussi eu l'occasion de recevoir des formations ou des indications de la part d'intervenants, internes au C.M.A. ou venant de l'INRIA :

- Merci tout particulièrement à Thierry PARMENTELAT, chercheur à l'INRIA, pour sa précieuse introduction aux Modèles de Markov Cachés et les nombreuses informations sans lesquelles le système de reconnaissance vocale n'aurait jamais vu le jour
- Laurent THÉRY, chercheur lui aussi à l'INRIA, qui nous a fait découvrir la preuve de programme en nous formant à l'utilisation du programme de preuve formelle Coq
- Annie RESSOUCHE de l'INRIA qui nous a fait découvrir le logiciel Scade Suite développé par Esterel Technologies, merci pour ses explications et les réponses à nos questions
- Brigitte HANOT, bibliothécaire au C.M.A
- Jean-Paul MARMORAT, chercheur au C.M.A., pour ses conseils avisés en traitement du signal

Table des matières

0.1	Présentation des enjeux	5
0.2	Objectifs du projet	5
0.3	Approches de la reconnaissance vocale	6
0.3.1	Acoustique-phonétique	6
0.3.2	Reconnaissance de motifs	7
I	Démarche Technique	8
1	Principe général du traitement du signal	8
1.1	Objectifs	8
1.2	Schéma global	8
2	Analyse, formatage du signal	10
2.1	Introduction	10
2.2	Prérequis	10
2.2.1	Qu'est-ce que le son ?	10
2.2.2	Comment le son est-il représenté dans l'ordinateur ?	10
2.3	Préparation du signal en vue du traitement	12
2.3.1	Synchronisation	12
2.3.2	Accentuation des hautes fréquences	12
2.4	Découpage du signal	14
2.5	Du temporel au fréquentiel	15
2.6	Simulation du comportement de l'oreille humaine	16
2.7	Retour en temporel	18
2.8	Obtention des coefficients caractéristiques du son	19
2.9	Schéma récapitulatif	19
3	Modélisation des mots à reconnaître par les modèles de Markov cachés	20
3.1	Objectifs	20
3.2	Prérequis et principe	20
3.2.1	Les automates	20
3.2.2	Les modèles de Markov cachés	20
3.2.3	Modèles discrets et modèles continus	21
3.2.4	Application à la reconnaissance vocale	22
3.3	Principaux algorithmes sur les modèles de Markov	22
3.4	Application à notre objectif	22
3.5	Phase d'apprentissage	23
3.6	Phase de reconnaissance	23
3.7	Récapitulatif du principe de création du modèle de Markov caché	24

II	Approche commerciale	26
1	Approche du développement du projet	26
1.0.1	Choix d'une architecture optimale pour notre projet	26
1.0.2	Réalisation du SpeechServer	28
1.0.3	Système de Gestion de Base de Données (SGBD)	29
1.1	Dimensionnement de l'infrastructure de calcul de The Speech App Company	29
1.1.1	Hypothèses de fonctionnement	29
1.1.2	Dimensionnement en mémoire RAM et espace disque	30
1.1.3	Dimensionnement réseau	30
1.1.4	Dimensionnement des éléments de calculs	30
1.1.5	Choix de l'infrastructure et coûts induits	30
1.1.6	SpeechRecorder	31
1.1.7	SpeechApp	32
2	Étude de marché et applications	35
2.1	Le marché actuel	35
2.2	Principaux domaines d'application	36
3	Budget, modèle économique	38
3.1	Introduction	38
3.2	Les salaires	38
3.3	Le compte de résultat prévisionnel	39
3.4	Le bilan	40
3.5	Les impôts	41
3.6	Conclusion et vue sur le long terme	41
III	Code	43
A	Code Principal	43
A.1	shell.py	43
A.2	server.py	48
A.3	gui.py	48
B	handling	53
B.1	fenetrehann.py	53
B.2	inverseDCT.py	53
B.3	triangularFilterbank.py	54
B.4	passerhaut.py	54
B.5	fft.cpp	55
C	HMM	58
C.1	creationVecteurHMM.py	58
C.2	markov.py	59
C.3	tableauEnergyPerFrame.py	62
C.4	hmm.cpp	62
D	recorder	77
D.1	recorder.py	77
D.2	sync.py	78

E	utils	81
E.1	animate.py	81
E.2	constantes.py	81
E.3	db.py	82
E.4	util.py	86
F	SpeechApp	86
F.1	main.js	86
F.2	index.html	90
G	SpeechServer	93
G.1	main.py	93
G.2	audioConverter.py	94
G.3	clientAuth.py	96
G.4	speechActions.py	98

Introduction

Dans le cadre du projet *Métiers de l'Ingénieur Généraliste - Systèmes embarqués* (projet d'enseignement en première année de l'école MINES ParisTech), nous nous sommes attaqués au sujet complexe et pluridisciplinaire du « traitement automatique de la parole ».

Ce document décrit comment 13 élèves-ingénieurs en première année à MINES ParisTech ne possédant, a priori, pas de compétences dans ce domaine, se sont appropriés ce sujet et ont réussi à l'approfondir à travers la réalisation complète d'un logiciel de reconnaissance vocale que nous avons baptisé SPEECHAPP.

0.1 Présentation des enjeux

La reconnaissance vocale automatisée est l'objet d'intenses recherches depuis plus de 50 ans. Malgré son caractère à première vue futuriste (souvent présentée dans des oeuvres de science-fiction), elle a pris sa place dans notre quotidien avec la prolifération de systèmes embarquant une telle technologie, comme par exemple le logiciel Siri dans les téléphones d'Apple[1]. Les perspectives économiques qui s'ouvrent au détenteur d'un système de reconnaissance fiable, robuste, et portable sont innombrables et l'on ne saurait surestimer son importance, (systèmes embarqués, commandes vocales, aide aux sourds/muets, etc.). Les systèmes les plus aboutis offrent des performances remarquables mais le problème reste toujours ouvert et suscite plus d'engouement que jamais en raison de la puissance croissante de calcul disponible, des dernières avancées technologiques et de la découverte de nouvelles applications.

La complexité de cette problématique s'explique notamment par la grande diversité des thèmes qui lui sont connexes et avec lesquels tout système se voulant performant se doit de traiter : traitement du signal, théorie de l'information, acoustique, linguistique, intelligence artificielle, physiologie, psychologie, etc. La reconnaissance vocale requiert en effet des connaissances extrêmement diverses ; la capacité à exploiter des ressources, qui ne nous sont pas de prime abord particulièrement familières, devient alors un atout capital. Le sujet traité ne se réduit pas à la seule détermination d'une suite de mots prononcés, mais peut s'étendre à divers autres applications telles que la reconnaissance de la langue, de l'accent d'un locuteur, voire la détermination de son sexe et de son âge, de s'il est stressé ou calme, de l'environnement il se trouve. Ces paramètres influent en effet de manière capitale l'analyse de la parole.

0.2 Objectifs du projet

Ce MIG s'est placé dans une perspective résolument plus humble en raison du temps imparti. Il ne s'agissait pas de réaliser un programme prétendant rivaliser avec les actuels systèmes de reconnaissance, fruits de nombreuses années de recherche et de développement, mais plutôt, à l'instar de l'ingénieur généraliste, de prendre connaissance d'un sujet et d'une problématique et de tâcher, en équipe, d'y apporter une solution qui soit la plus optimale possible compte tenu des contraintes temporelles et matérielles. Le projet des MIG ne se réduisant pas non plus à une réalisation technique, il s'agissait de garder en vue les perspectives économiques et les composantes juridiques, indissociables d'un tel projet, comme garde-fou de toute pérégrination informatique.

De plus, ce système de reconnaissance vocale, qui peut sembler immédiat, voire intuitif, tel qu'on l'expérimente aujourd'hui, n'est en fait pas aussi évident qu'on pourrait le croire. Cela se manifeste

notamment dans le fait qu’aucun système de reconnaissance automatique de la parole n’atteigne aujourd’hui un taux de réussite de cent pour cent. Nombreux sont ceux qui ont déjà ressenti la frustration de ne pas être compris par la machine sensée reconnaître nos paroles. Il convient donc de préciser ce qui rend la tâche si subtile face à ce que nos oreilles et notre cerveau font instantanément.

Les rôles pour le développement du logiciel ont été attribués dès le début selon les goûts et compétences de chacun. Cela n’a empêché personne, grâce à la répartition des tâches et à la diversité intrinsèque au projet, d’exploiter un panel très diversifié de compétences tout en apportant la valeur ajoutée de sa spécialité. Dans la chaîne du développement, chaque fonction dépendant très fortement de celles qui la précèdent et de celles qui la suivent, une bonne communication interne était indispensable pour un développement cohérent et efficace. Il nous a été nécessaire de coordonner notre activité pour s’assurer que nos fonctions s’enchaînaient sans erreur : cela implique que l’un d’entre nous a pris le rôle de coordinateur pendant tout le projet. Par exemple, l’utilisation de la plateforme GitHub[2] pour l’échange de fichiers et de mises à jour s’est révélée particulièrement efficace et permettait à chacun d’incorporer en temps réel ses dernières modifications sans empiéter sur le travail des autres. La complexité de la discipline fut un des principaux obstacles, et une phase d’appropriation des techniques requises, de par la lecture de livres dédiés, d’articles de recherches ainsi que de thèses a été le poumon du projet.

0.3 Approches de la reconnaissance vocale

Avant de rentrer dans des considérations techniques, il est nécessaire de définir un principe d’étude, une stratégie de résolution qui dictera l’orientation générale du projet en plus de rendre les objectifs et les enjeux plus clairs. Cette partie a pour but de donner un aperçu des différents angles d’attaque du problème à considérer, ainsi que de présenter celui que nous avons choisi, et les raisons qui ont amené à ce choix.

Dans son livre *Fundamentals of speech recognition*, Lawrence Rabiner[3] dégage des travaux de ces prédécesseurs trois approches conceptuelles du problème : l’approche acoustique-phonétique, l’approche par reconnaissance de motifs et l’approche par intelligence artificielle. Cette dernière n’étant, d’après Rabiner, qu’un avatar de la première, nous ne présenterons que l’acoustique phonétique et la reconnaissance de motifs que nous avons choisie pour notre projet.

0.3.1 Acoustique-phonétique

L’approche acoustique-phonétique est indubitablement celle qui paraît la plus naturelle et directe pour faire de la reconnaissance vocale et est celle qui s’impose à priori à l’esprit. Le principe est le suivant : l’ordinateur tâche de découper l’échantillon sonore de manière séquentielle en se basant sur les caractéristiques acoustiques observées et sur les relations connues entre caractéristiques acoustiques et phonèmes. Ceci dans le but d’identifier une suite de phonèmes¹ et d’ainsi reconnaître un mot.

Cette approche suppose qu’il existe un ensemble fini de phonèmes différenciables et que leurs propriétés sont suffisamment manifestes pour être extraites d’un signal ou de la donnée de son spectre (tableau des fréquences et de leur amplitude associée, composant un signal à un instant donné) au cours du temps. Même s’il est évident que ces caractéristiques dépendent très largement du sujet parlant, nous partons du principe que les règles régissant la modification des paramètres peuvent être apprises et appliquées.

1. « En phonologie, domaine de la linguistique, un phonème est la plus petite unité discrète ou distinctive (c’est-à-dire permettant de distinguer des mots les uns des autres) que l’on puisse isoler par segmentation dans la chaîne parlée. Un phonème est en réalité une entité abstraite, qui peut correspondre à plusieurs sons. Il est en effet susceptible d’être prononcé de façon différente selon les locuteurs ou selon sa position et son environnement au sein du mot. » (Définition Wikipédia du mot phonème)

Bien que cette méthode ait été vastement étudiée et soit viable, on lui préférera l'approche par reconnaissance de motifs qui, pour plusieurs raisons, l'a supplantée dans les systèmes appliqués. C'est celle que nous avons choisi et que nous présentons dans le prochain paragraphe.

0.3.2 Reconnaissance de motifs

Cette technique diffère de la méthode précédente par le fait qu'elle ne cherche pas à exhiber des caractéristiques explicites. Elle se compose de deux étapes : « l'entraînement » des motifs, et la reconnaissance via la comparaison de ces motifs.

L'idée sous-jacente au concept d'entraînement repose sur le principe selon lequel si l'on dispose d'un ensemble suffisamment grand de versions d'un motif à reconnaître, on doit être capable de caractériser pertinemment les propriétés acoustiques du motif. Notons que les motifs en question peuvent être de nature très diverses, comme des sons, des mots, des phrases, ce qui sous-tend l'idée d'un grand nombre d'applications théoriques comme présenté en introduction. La machine apprend alors quelles propriétés acoustiques sont fiables et pertinentes. On effectue ensuite une comparaison entre le signal à reconnaître et les motifs tampons, afin de le classer en fonction du degré de concordance.

Sans plus entrer dans les détails, les avantages de cette approche qui nous ont poussés à l'adopter sont les suivants :

- Elle est simple à appréhender, et est très largement comprise et utilisée
- Elle est robuste, c'est-à-dire qu'elle dépend peu du locuteur et de l'environnement
- Elle donne lieu à de très bons résultats

Première partie

Démarche Technique

Avant de commencer un projet informatique d'une telle envergure, il faut faire des choix techniques. Dans un premier temps, nous avons décidé de programmer le projet en **Python**². En effet, ce dernier est facile à prendre en main, permet une programmation rapide et efficace et dispose d'un grand panel de bibliothèques bien documentées. En ce qui concerne ces dernières, nous avons utilisé :

- pyaudio³ pour l'écriture et la lecture des fichiers audio .wav⁴
- numpy⁵ et scipy⁶ pour faire des mathématiques avancées non incluses dans la bibliothèque standard tels que du calcul numérique de haute précision et du calcul matriciel

Néanmoins, nous nous sommes vite rendu compte que le langage Python était lent. Or, notre programme s'est montré être gourmand en ressource processeur. Nous avons donc fait le choix d'implémenter certaines fonctions en C++, langage nettement plus rapide. Enfin, afin de coordonner nos efforts et de permettre une meilleure répartition des tâches, nous avons fait le choix d'utiliser les services du site GitHub[2] basé sur git⁷.

1. Principe général du traitement du signal

1.1 Objectifs

Bien que la reconnaissance vocale telle qu'elle est aujourd'hui mise-en-place dans les différents matériels semble immédiate, le travail à effectuer pour reconnaître un mot est complexe. La première étape pour faire de la reconnaissance vocale est de parvenir à trouver un moyen de caractériser efficacement et uniformément un mot. Cela signifie désigner un mot par un certain motif puis permet par le même procédé appliqué sur un enregistrement quelconque, de parvenir à identifier le motif le plus vraisemblable qui correspondrait alors au même mot. Il s'agit donc tout d'abord de traiter le signal pour en découvrir certaines caractéristiques. En effet, une même personne ne prononce pas toujours les mots de la même façon, au même débit, avec les mêmes hauteurs de son, ce qui rend ardue une simple identification par comparaisons temporelles.

1.2 Schéma global

Afin de gérer ces difficultés, nous avons mis en place plusieurs étapes successives de traitement supplémentaire afin d'obtenir cette fameuse « trace » qui caractériserait un enregistrement, c'est-à-dire un mot. Nous avons pour cela utilisé plusieurs techniques de traitement du signal communément reconnues (échantillonnage, fenêtrage, transformée de Fourier directe et inverse). Cette figure explique

2. www.python.org

3. <http://people.csail.mit.edu/hubert/pyaudio/>

4. WAV (ou WAVE), une contraction de WAVEform audio file format, est un standard pour stocker l'audio numérique de Microsoft et IBM. (Wikipédia)

5. www.numpy.org

6. www.scipy.org

7. Git est un logiciel de gestion des versions décentralisé

globalement le traitement que nous avons choisi de mettre-en-place afin de reconnaître le mot prononcé. Il y a donc plusieurs étapes qui s'enchaînent pour parvenir à un objet que nous pourrions manipuler en le sachant représentatif et caractéristique du son.

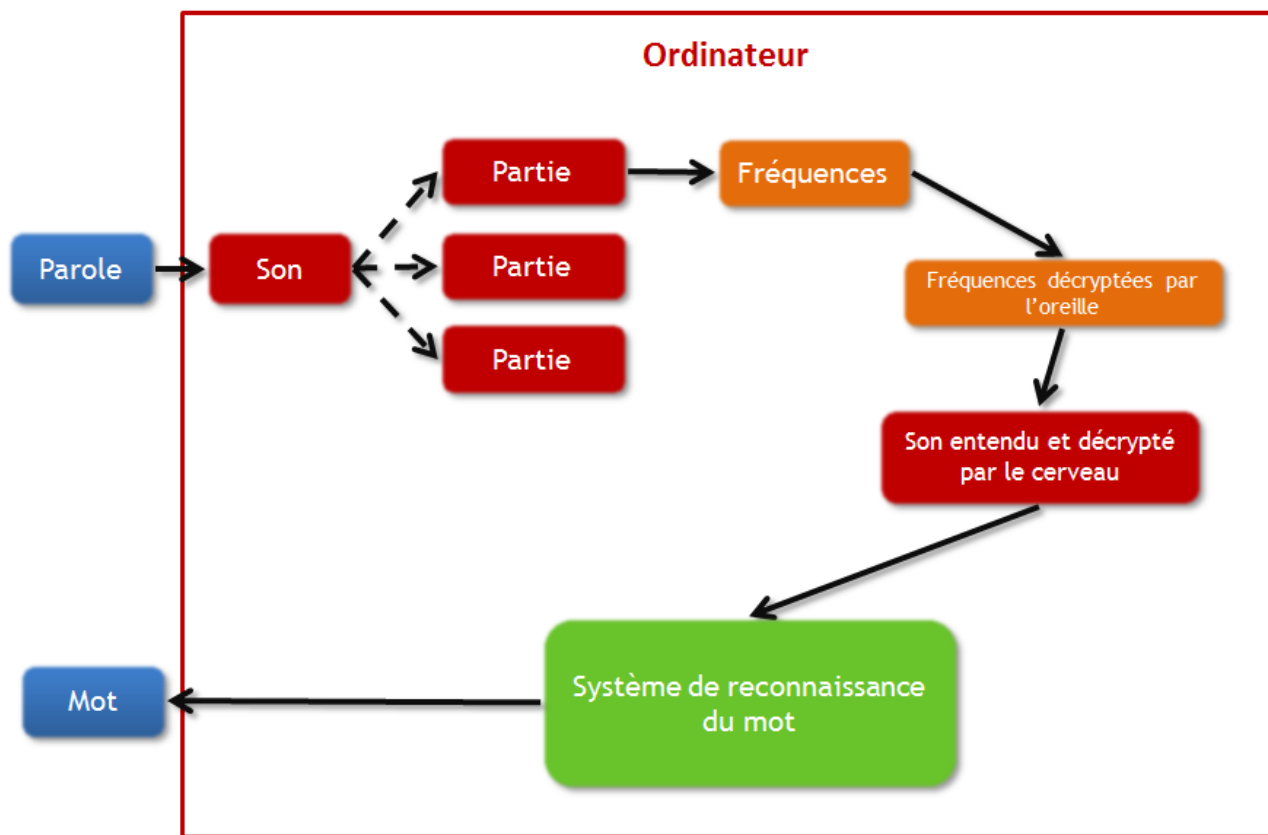


FIGURE 1.1 – Traitement du son pour le reconnaître

Enregistrement du son La première étape consiste simplement à enregistrer le son sur le disque dur de l'ordinateur. Nous utilisons pour cela le module intégré à Python appelé PyAudio[4]. Cela permet d'enregistrer avec une certaine *fréquence d'échantillonnage* (donc un certain nombre de captures de son par seconde) les amplitudes du son captées par le micro.

Découpage en fenêtre Le son est découpé ensuite en petites fenêtres de quelques dizaines de millisecondes ce qui permet d'isoler les événements sonores qui pourraient avoir une importance. Il s'agit d'un *fenêtrage*.

Passage en fréquence Jusque là, le son étudié représentait temporellement ce qui avait été entendu. Néanmoins, il est difficile d'étudier un son tel quel et on utilise alors le lien entre les fréquences et le signal temporel. Il est ensuite plus facile d'étudier et de transformer un ensemble de fréquences pour appliquer par un exemple des filtres qui rapprochent le programme du fonctionnement de l'oreille.

Utilisation de l'échelle de Mel Puisque le programme doit savoir *faire la différence entre des mots*, c'est-à-dire des sons identifiés tels quels par une oreille *humaine*, il faut donner au programme un comportement similaire à celui d'une oreille humaine. On utilise pour cela une échelle qui accentue certaines fréquences. En effet, il a été montré[5] (et ensuite appliqué [6]) que l'oreille ne perçoit pas toutes les fréquences de la même façon.

2. Analyse, formatage du signal

2.1 Introduction

Comme nous l'avons mentionné, même le plus élémentaire des systèmes de reconnaissance vocale utilise des algorithmes au carrefour d'une grande diversité de disciplines : reconnaissance de motifs statistiques, théorie de l'information, traitement du signal, analyse combinatoire, linguistique entre autres, le dénominateur commun étant le traitement du signal qui transforme l'onde acoustique de la parole en une représentation paramétrique plus adaptée à l'analyse automatisée. Le principe est simple : garder les traits distinctifs du signal et éviter au maximum tout ce qui pourra en parasiter l'étude. Cette conversion ne se fait donc pas sans perte d'information, et la délicatesse de la discipline tient en la sélection judicieuse des outils les plus adaptés afin de trouver le meilleur compromis entre perte d'information et représentation fidèle du signal.

2.2 Prérequis

2.2.1 Qu'est-ce que le son ?



FIGURE 2.1 – Oreille humaine

Le son est une onde mécanique se traduisant par une variation de la pression au cours du temps. Cette onde est caractérisée par différents facteurs comme son amplitude à chaque instant, qui est en d'autres termes la valeur de la dépression à cet instant, et par les fréquences qui la composent et qui changent au cours du temps.

2.2.2 Comment le son est-il représenté dans l'ordinateur ?

En se propageant, l'onde mécanique qu'est le son fait vibrer la membrane du micro. L'amplitude de la vibration dépend directement de l'amplitude du son. La position de la membrane est enregistrée à

intervalles de temps réguliers définis par l'échantillonnage.

L'échantillonnage correspond au nombre de valeurs prélevées en une seconde (principe [7]). Par exemple un échantillonnage à 44100 Hz correspond à relever la position de la membrane 44100 fois par secondes. La valeur de la position de la membrane est alors enregistrée sous la forme d'un entier signé codé sur n bits (n valant généralement 8,16,32 ou 64). Plus n est grand, plus la position de la membrane sera représentée de manière précise, et donc plus la qualité du son sera bonne. Grâce à l'échantillonnage et à la position de la membrane (n), on définit aisément le *bitrate*, qui correspond au débit d'information par seconde, de la façon suivante : $bitrate = n * \text{échantillonnage}$.

Ce dont nous disposons donc pour analyser un signal, est la donnée de l'amplitude en fonction du temps la caractérisant.

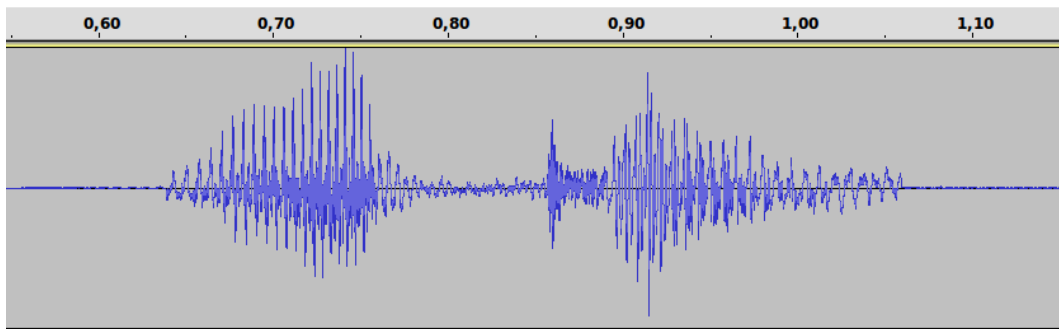


FIGURE 2.2 – Exemple audiogramme prononciation du mot "VICA"

2.3 Préparation du signal en vue du traitement

2.3.1 Synchronisation

Afin de synchroniser le début des enregistrements d'un mot, et de leur donner la même durée, nous avons eu l'idée de détecter les silences avant et après le mot pour les couper.

Le signal est lissé à l'aide d'une moyenne sur plusieurs échantillons pour que les fluctuations inhérentes à l'enregistrement ne gênent pas notre fonction. On détecte alors le moment où le signal (en valeur absolue) dépasse pour la première fois une valeur seuil et celui à partir duquel le signal ne dépasse plus celle-ci.

On sait alors où couper le signal d'origine, en élargissant légèrement la coupe afin d'éviter de supprimer des consonnes peu sonores. Cela permet en plus d'afficher un message d'erreur suspectant un enregistrement ayant commencé trop tard ou fini trop tôt. Deux problèmes se posent : en pratique, un bruit trop important perturbe le signal et le mot n'est plus détectable par l'amplitude des oscillations.

Toutefois, pour l'enregistrement de notre base de données, une pièce calme et un micro de bonne qualité nous ont permis un découpage satisfaisant, ce qui ne résout pas définitivement le problème, l'utilisateur ne pouvant pas toujours se placer dans ces conditions, le signal est traité par un filtre anti-bruit.

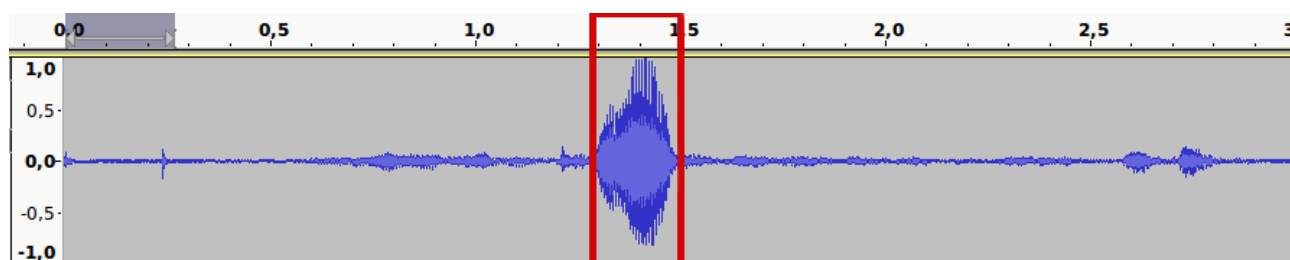


FIGURE 2.3 – Ensemble du son enregistré, la partie nous intéressant (le mot) est encadrée en rouge

Ce filtre consiste en l'utilisation de bibliothèques, SoX et ffmpeg([8] et [9]), qui permettent par l'étude d'un court laps de temps de bruit de soustraire le bruit de l'enregistrement. Nous n'avons pas cherché à traiter nous-même le bruit car il s'agit d'un problème complètement à part et qui ne demande pas les mêmes compétences que le traitement du signal effectué jusque là.

De plus, il a fallu déterminer la valeur de nos constantes de découpe (coefficient de lissage, coefficient de coupe, intervalle de temps de sécurité), qui dépendent bien sûr les uns des autres. Ceci a été fait de manière empirique sur plusieurs enregistrements de mots différents, permettant une découpe automatique la plus satisfaisante possible pour l'ensemble des mots.

2.3.2 Accentuation des hautes fréquences

En temps normal, malgré la présence d'un bruit constant plus ou moins envahissant, le cerveau humain est capable de décrypter le signal sonore perçu en faisant une distinction très précise entre le signal réel et les perturbations ambiantes. Toutefois cette capacité hors norme est propre au cerveau, et un système de reconnaissances en est initialement incapable. Il faut donc tenter de trouver une méthode algorithmique qui sépare au mieux le bruit ambiant du signal réel et ce dans le but d'optimiser au maximum les performances du système.

En effet, les performances de tout système de reconnaissances dépendent fortement de la variabilité des données (locuteur, environnement, bruit, réverbération, ...). Plus ces données sont variables, plus le taux d'erreur sera grand et un système de reconnaissance qui se veut être utilisable dans la vie de tous les jours : (voiture, endroits bruyants) ; se doit d'y remédier. Ces effets se font particulièrement sentir dans les basses fréquences, c'est pourquoi le conditionnement du signal en vue de son étude comprend inmanquablement un filtre passe haut c'est-à-dire une accentuation des amplitudes associées aux hautes fréquences et une diminution des basses fréquences. C'est le même principe qui est utilisé dans les égaliseurs des lecteurs de musique d'aujourd'hui qui proposent d'augmenter les basses ou les aigus. Les filtres passe-haut améliorent significativement les résultats de reconnaissance comme en témoignent les expériences de H.G. Hirsch P. Meyer et H.W. Ruehl dans leur papier[10].

Utiliser un filtre passe-haut présente comme avantage de ne pas nécessiter de procéder au préalable à une reconnaissance de silence contrairement aux techniques de réduction du bruit et de soustraction spectrale.

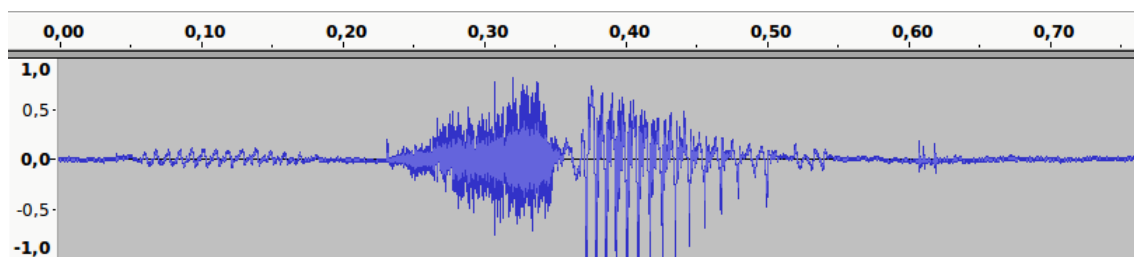


FIGURE 2.4 – Exemple d'un fichier son (représentant « Cinq ») **avant** application du filtre

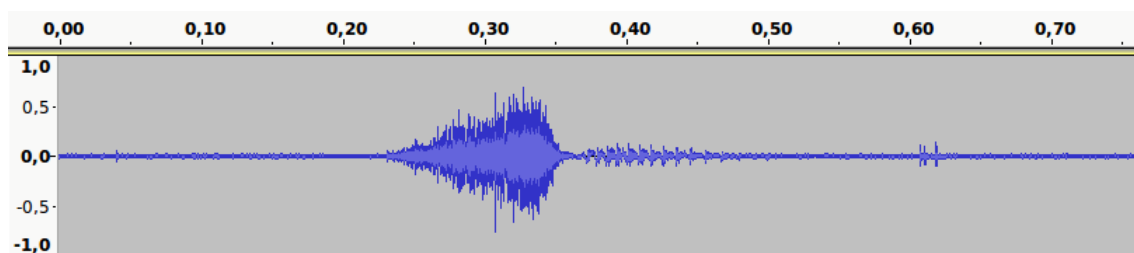


FIGURE 2.5 – Exemple du fichier son **après** application du filtre

Le signal étant caractérisé par une suite (x_n) d'amplitudes, comme présenté dans les prérequis, où n représente un instant de la musique déterminé par l'échantillonnage ; on opère linéairement la transformation suivante sur le signal : $y_0 = x_0$ et $y_n = x_n - 0.95 * x_{(n-1)}$ pour $n > 0$, où y représente le signal de sortie après transformation.

Cette opération consiste effectivement en un filtre passe-haut, en effet une telle formule part du principe que 95% d'un échantillon a pour origine l'échantillon précédent. Ce constat étant plus pertinent pour les hautes fréquences (car les pics de l'onde associée sont plus rapprochés et engendrent donc un pic d'amplitude plus régulièrement), l'influence des basses fréquences est donc discriminée.

Cependant, contrairement au filtre passe-haut, le traitement du son enregistré ne se fait jamais sur la totalité du signal. Le signal est en fait découpé en petits bouts appelés échantillons sur lesquels les transformations seront appliquées. On obtient alors en guise de représentation du signal une suite de vecteurs caractérisant chacun un morceau de signal. C'est ce découpage que la partie suivante présente.

2.4 Découpage du signal

L'analyse du signal repose sur l'étude des fréquences présentes à un instant donné ce qui ne peut se faire si l'on considère le signal dans sa totalité car l'on aurait alors une moyenne sur toute la piste des fréquences présentes et non un ensemble de données ponctuelles. Le procédé que nous avons mis en place pour pallier à ce problème est celui le plus couramment utilisé dans ce domaine : l'échantillonnage. Nous avons découpé le signal à traiter en petites séquences, qui, juxtaposées, approximent une échelle temporelle continue.

La taille des échantillons est un paramètre déterminant sur la qualité et la précision de l'analyse combinée finale. Une fois calculé, le spectre ne reflète plus du tout de dépendance temporelle. La durée d'un échantillon correspond ainsi à la durée minimale d'un événement sonore détectable. Il faut donc réduire cette durée autant que possible, pour obtenir une discrétisation temporelle la plus proche possible de la continuité. Il est en revanche nécessaire de conserver un certain nombre de points par échantillon. En effet, le spectre obtenu par l'analyse sera plus précis et proche de la réalité fréquentielle si le nombre de points du signal analysé est important. La meilleure technique pour contourner ce compromis est d'augmenter la fréquence d'échantillonnage. On obtient alors un nombre important de points qui s'étirent peu dans le temps.

Le théorème de Nyquist-Shannon[11] assure qu'un signal reproduit fidèlement toutes les fréquences inférieures à la moitié de sa fréquence d'échantillonnage. Une fréquence d'échantillonnage de 44100Hz (parfois 48000Hz) est donc suffisante pour couvrir la totalité d'une oreille humaine en bonne santé. L'utilisation la plus courante de l'enregistrement audio étant (à notre niveau) la restitution, le matériel et le logiciel à notre disposition se cantonnaient à ces fréquences d'échantillonnage. Nous avons ainsi dû trouver un compromis entre résolution fréquentielle et précision temporelle. L'hypothèse principale a été que les événements sonores et variations s'étalant sur une durée inférieure à 20 millisecondes n'étaient pas significatifs pour notre analyse. Le nombre de points a donc été couplé à notre fréquence d'échantillonnage lors des enregistrements, à 44100Hz.



FIGURE 2.6 – Principe normal du fenêtrage



FIGURE 2.7 – Fenêtrage de Hann évitant les discontinuités

L'échantillonnage introduit par ailleurs des discontinuités aux bornes des morceaux, qui ne sont pas présentes dans le signal original. Le fenêtrage permet de réduire l'effet de ces discontinuités virtuelles. On découpe le signal en plus de morceaux, tout en conservant la même durée pour chaque échantillon. On obtient des "fenêtres", qui se recoupent les unes les autres. Pour que la même partie du signal ne soit pas retraitée à l'identique, on applique une fonction - dite fonction de fenêtrage, ou dans notre cas, fonction de Hann - qui diminue l'importance des valeurs situées aux extrémités de la fenêtre. Ce procédé a le désavantage de démultiplier le temps de calcul des étapes suivantes de l'algorithme (le nombre d'échantillons est bien plus important pour un signal de même longueur). Certaines applications (notamment pour les téléphones portables) devant réduire la complexité au maximum en font donc abstraction. Notre reconnaissance privilégiant plutôt la précision, et disposant d'une puissance de calcul largement suffisante pour conserver un rendu de l'ordre de la seconde, nous avons opté pour un fenêtrage important (recouvrement total d'un échantillon par ses voisins), au prix d'une multiplication du temps de calcul par deux.

Maintenant que nous avons découpé notre signal en petits morceaux, voyons comment sur chaque morceau nous pouvons obtenir les fréquences présentes. Du fait que ces morceaux représentent une durée très petite, on peut considérer que les fréquences obtenues ne sont plus une moyenne sur une longue période mais bien des fréquences présentes à un instant précis.

2.5 Du temporel au fréquentiel

Le domaine temporel est parfait pour l'acquisition et la restitution de l'audio, car il représente fidèlement la vibration de la membrane d'un micro ou d'une enceinte. L'oreille humaine base sa perception et sa reconnaissance sur le domaine fréquentiel. Il faut donc passer de l'un à l'autre, et ce grâce à l'utilisation de la transformation de Fourier. L'algorithme "intuitif" de calcul ayant, pour trouver le spectre d'un unique échantillon, une complexité en $O(N^2)$ (avec N le nombre de points par échantillons), il est nécessaire de trouver d'autres méthodes si l'on envisage des applications proches du temps réel. Heureusement, plusieurs approches se sont ouvertes à nous pour l'optimisation du temps de calcul.

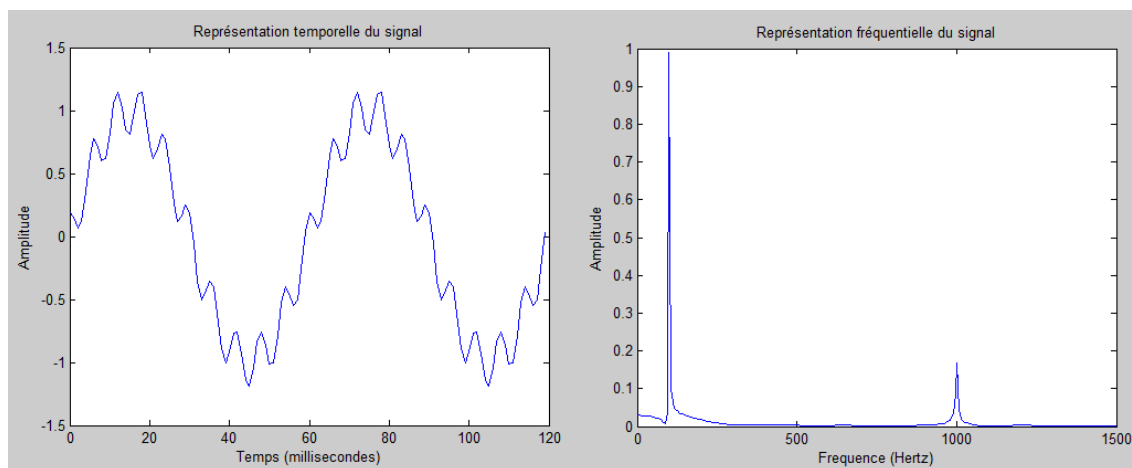


FIGURE 2.8 – Exemple de passage du domaine temporel (somme de cosinus) au domaine fréquentiel (pics pour les fréquences fondamentales)

Le calcul de la transformée de Fourier est incontournable en analyse du signal, et il a donné lieu à de nombreuses études. Des algorithmes optimisés pour diverses utilisations sont disponibles, et notre travail a surtout été d'identifier lequel s'adapterait à notre projet. La fonction que nous avons implémentée est l'algorithme de Cooley-Tukey, qui permet de réduire la complexité à $O(N \log_2(N))$, et qui repose sur le fonctionnement diviser pour régner. Le principe est dans un premier temps de diviser le signal à analyser en sous-tableaux de mêmes tailles, de manière croisée (par exemple deux sous-tableaux, pour

les indices pairs et impairs). On calcule ensuite les transformées de Fourier de ces sous-tableaux, en opérant récursivement, jusqu'à obtenir des sous-tableaux dont la taille est un entier. On calcule leur transformée de Fourier, et on recombine les résultats obtenus. Cette méthode a l'avantage de pouvoir être couplée à d'autres algorithmes pour calculer les spectres des sous-tableaux dont la taille n'est pas un produit d'entier. Le meilleur cas est alors instinctivement un signal initial dont la longueur est une puissance de deux. Il est même intéressant d'utiliser la technique du bourrage de zéros (zero padding), qui consiste à rajouter des zéros à la suite du signal pour atteindre la puissance de deux la plus proche. Cela ne change pas le spectre obtenu et augmente les performances. Dans notre cas, nous avons eu la possibilité d'ajuster la taille des échantillons. Nous avons ainsi choisi des échantillons de 1024 points, ce qui correspond, avec notre fréquence d'échantillonnage de 44100Hz, à une durée d'environ 23ms. Seul le dernier échantillon du signal est complété par des zéros.

De plus, comme les données sur lesquelles nous travaillons sont réelles, et que les calculs de la Transformée de Fourier Rapide (Fast Fourier Transform, ou FFT) s'effectuent avec des complexes, la première idée d'optimisation que nous avons eue est de calculer le spectre de deux échantillons à la fois, en créant des complexes à partir des deux signaux réels (l'un représente la partie réelle, l'autre imaginaire). On obtient rapidement les coefficients respectifs des deux échantillons par une simple opération sur le spectre résultant. Cependant, cette méthode ne divise le temps de calcul que par deux, et notre FFT demeure trop lente (plusieurs secondes pour un signal d'environ une seconde), surtout au regard du temps de calcul total de la reconnaissance en elle-même. La deuxième optimisation que nous avons donc appliquée est de passer le code du Python au C++, langage compilé beaucoup plus rapide. De plus, nous avons repensé les fonctions, de façon à éviter les appels récursifs. En effet, le travail sur des tableaux force une recopie à chaque appel de fonction, ce qui démultiplie la complexité du calcul. Le résultat est un algorithme qui s'effectue en moins d'une seconde, et qui peut s'inscrire dans un contexte d'exploitation en temps réel.

2.6 Simulation du comportement de l'oreille humaine

Des études de psycho acoustique ont montré que l'oreille humaine ne percevait pas les fréquences selon une échelle linéaire[5]. Il a donc été utile de définir une nouvelle échelle plus subjective : à chaque fréquence f , exprimée en Hertz, on fait correspondre une nouvelle fréquence selon une fonction censée représenter le comportement de l'oreille humaine. Par convention, la fréquence de 1000 Hz correspond à 1000 mel. Les autres fréquences mel sont ajustées de façon à ce qu'une augmentation de la fréquence mel corresponde à la même augmentation de la tonalité perçue. Cela conduit à la fonction *mel* suivante :

$$mel(f) = 2595 * \log(1 + f/700)$$

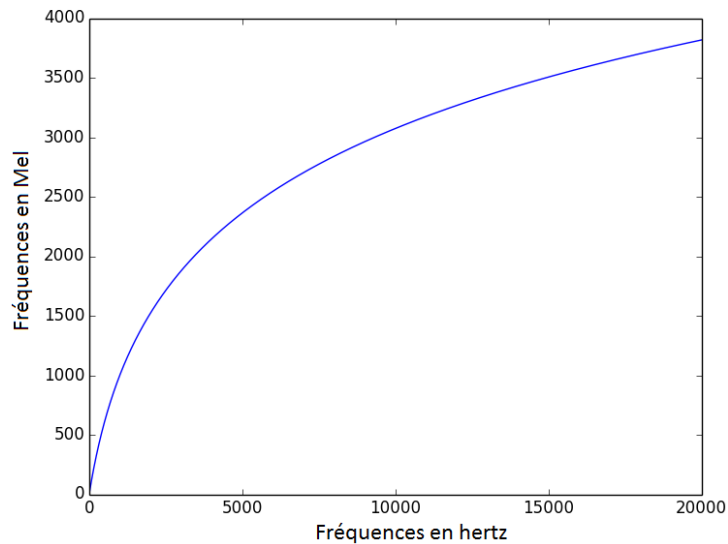


FIGURE 2.9 – Graphe de conversion

On remarque que le poids des hautes fréquences (supérieures à 1000 Hz) est diminué tandis que le poids des basses fréquences (inférieur à 1000 Hz) est augmenté.

Il est préférable d'employer cette échelle de fréquence dans l'algorithme de reconnaissance : ce dernier doit en effet différencier plusieurs mots selon la perception humaine, c'est-à-dire en simulant le comportement de l'oreille humaine.

Le ré-étalonnage en échelle Mel est réalisé en appliquant au signal des filtres triangulaires comme présentés dans la figure ci-dessous. On remarque qu'en effet, plus la fréquence est grande, plus le filtre est large ce qui signifie que l'on considère un plus grand intervalle de fréquence comme une seule et même information. Ceci illustre ce qui était précédemment dit sur la discrimination des hautes fréquences. Cette largeur de filtre est dictée par l'échelle Mel. Ceci conduit à l'obtention d'un tableau de 24 cases dont chaque case contient l'amplitude associée à l'intervalle de fréquence représenté par l'indice du tableau (et qui est comme on l'a dit, déterminé par l'échelle Mel).

Ce tableau qui caractérise donc un échantillon du signal, n'est pas utilisé tel quel dans la suite. En effet, il est utile pour des raisons qui seront précisées plus tard, de ne plus considérer un tableau de 24 cases échelonné en fréquences, mais en temporel ! La transformée en cosinus inverse opère cette transformation et est présentée dans la section suivante.

2.7 Retour en temporel

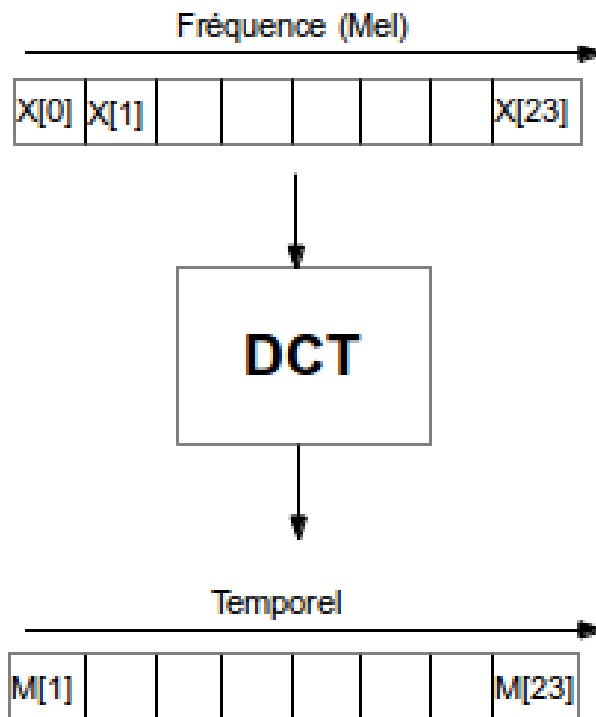


FIGURE 2.10 – Graphe de conversion

Dans les parties précédentes nous avons vu comment, à partir d'un extrait sonore échantillonné à 44100 Hz sur 16 bits, obtenir après transformée de Fourier et opérations sur le spectre, un tableau de 24 cases gradué en échelle Mel, représentant une fraction de l'extrait. Ce tableau exprimé ainsi en fréquences, pourrait a priori constituer une représentation satisfaisante de l'extrait sonore à l'instant considéré pour la suite de l'algorithme de reconnaissance, et servir à la comparaison avec le modèle au travers des chaînes de Markov cachées. Cependant ce n'est pas ce qui est fait, et l'on préférera une représentation en temporel de la fraction sonore considérée, ceci pour deux principales raisons.

- Opérer une transformation en cosinus inverse "décorrèle" les valeurs du tableau dans la mesure où dans une représentation en fréquentiel, les valeurs associées aux hautes fréquences sont très fortement corrélées avec ce qui se passe dans les basses fréquences. En effet, un signal sonore n'est jamais pur, c'est-à-dire constitué d'une seule fréquence, mais est un amalgame de signaux purs de fréquences multiples de celles d'autres signaux purs.

Le tableau qui sera traité par la suite grâce aux chaînes de Markov n'est plus constitué de 24 cases mais de 12 dont les 11 premières sont les premières cases du tableau obtenu après DCT. Si l'on tronquait le tableau avant d'opérer la DCT, on ne conserverait que l'information associée aux graves ce qui constituerait une perte trop importante de données.

- Ce retour au temporel se fait par la transformée en cosinus inverse. Il s'agit en terme simplistes du pendant réel de la transformée de Fourier inverse, qui elle donne lieu à des coefficients complexes, lesquels dans le cadre d'une représentation temporelle n'ont que peu de sens. En termes plus mathématiques, la projection orthogonale du signal discret en fréquentiel ne se fait plus sur une base d'exponentielles complexes, mais de cosinus.

La DCT que nous avons utilisé, aussi connue sous le nom de DCTII se base sur la formule suivante :

$$M[k] = \sum_{n=0}^{B-1} (X[n] \times \cos(\pi \cdot k \cdot \frac{n+0.5}{B})) \times \sqrt{\frac{2}{B}}$$

avec B=24, X le tableau en échelle Mel, et M le tableau de sortie échelonné en temporel. D'autres formules équivalentes de DCT existent mais la DCTII est la plus largement répandue et utilisée.

2.8 Obtention des coefficients caractéristiques du son

À ce stade nous disposons donc d'une séquence de tableaux de 24 cases, échelonnée en fréquences. La construction des **Mel Frequency Cepstral Coefficients** (aussi appelé **MFCC**) à partir de ces tableaux est un jeu d'enfant[6]. Il suffit pour chaque tableau de former un nouveau tableau composé de ses 11 premières valeurs et de rajouter comme 12ème valeur, la donnée de l'énergie du signal à l'instant considéré.

Cette recette s'appuie sur les deux constats suivants. Premièrement, en ne gardant que les 11 premières valeurs on ne se concentre que sur les fréquences graves ce qui rejoint ce qui était expliqué précédemment sur l'intérêt réduit des aigus. Deuxièmement, la donnée de l'énergie du signal est d'un intérêt majeur, au même titre que les fréquences présentes, car c'est un trait caractéristique d'un instant du signal qu'il ne faut pas négliger dans l'étude et il est important d'en garder la trace.

Nous disposons désormais de notre caractérisation adéquate du signal, il est temps de passer à la reconnaissance.

2.9 Schéma récapitulatif

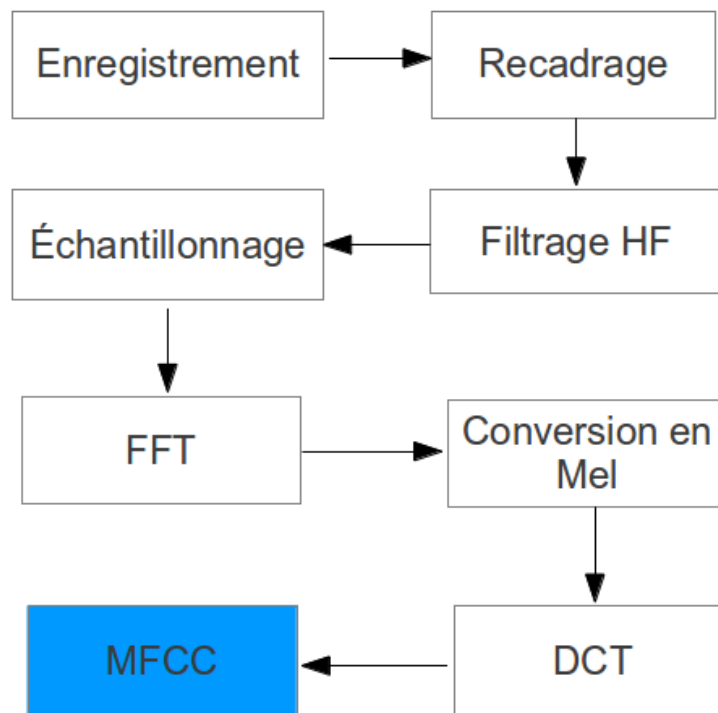


FIGURE 2.11 – Trajet du son dans notre programme

3. Modélisation des mots à reconnaître par les modèles de Markov cachés

3.1 Objectifs

Le traitement effectué sur le son permet d'obtenir un tableau fréquentiel caractéristique du mot. Nous pouvons alors rentrer dans le vif du sujet et travailler sur la reconnaissance même des mots : il faut parvenir à comparer entre eux les empreintes ainsi obtenues. Pour cela, nous utilisons un système appelé les *modèles de Markov cachés*. Nous créons en fait un « graphe de sons » qui peut être parcouru à partir d'un mot. En appliquant un certain algorithme à un mot et à un graphe, on obtient une probabilité qui témoigne de l'adéquation du mot au graphe. Il nous suffit alors de déterminer quel graphe représente la plus grande probabilité ce qui nous donne la solution comme étant le *meilleur candidat*. On parle alors de **maximum de vraisemblance**.

3.2 Prérequis et principe

Un modèle de Markov caché est un modèle statistique qui peut modéliser des processus physiques. Il fait appel aux structures d'automates[12].

3.2.1 Les automates

Un automate représente un système physique. Il est composé d'états (les cercles sur la figure), qui correspondent aux états du système réel, et de transitions (les flèches sur la figure), pour passer d'un état à l'autre. Il existe aussi la notion de chemin : par exemple pour passer de 0 à 3 sur la figure, il faut passer par 1 puis 2 : le chemin de 0 à 3 est 0,1,2,3.

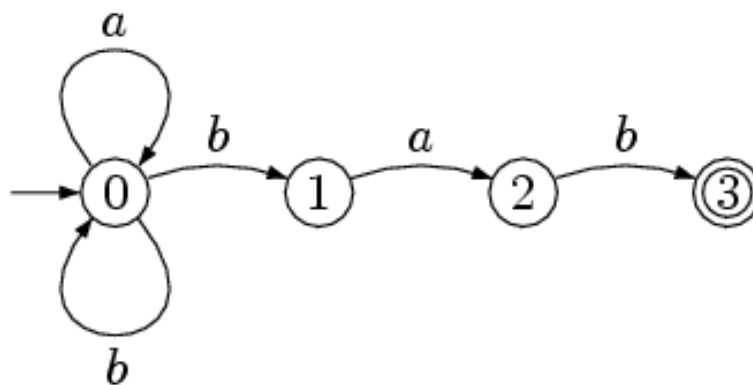


FIGURE 3.1 – Exemple d'automate « classique »

3.2.2 Les modèles de Markov cachés

Un modèle de Markov est un automate présentant *deux caractéristiques* en plus du principe de base d'un automate.

Tout d'abord, les transitions ne sont plus déterministes comme elles le sont dans le cas d'un automate mais sont **probabilistes**. Ainsi, une fois dans un état, on se dirige vers un autre état selon un arc avec une certaine probabilité : la probabilité de transition.

La seconde différence est que les données renvoyées lors du parcours d'un chemin de l'automate ne sont plus la liste des états par lesquels passe le chemin : chaque état a maintenant des probabilités d'émettre certains signaux. On obtient donc une liste de signaux pour un chemin. Or, un état peut émettre plusieurs signaux différents et l'émission n'est que probabiliste ce qui rajoute au caractère non déterministe de l'automate.

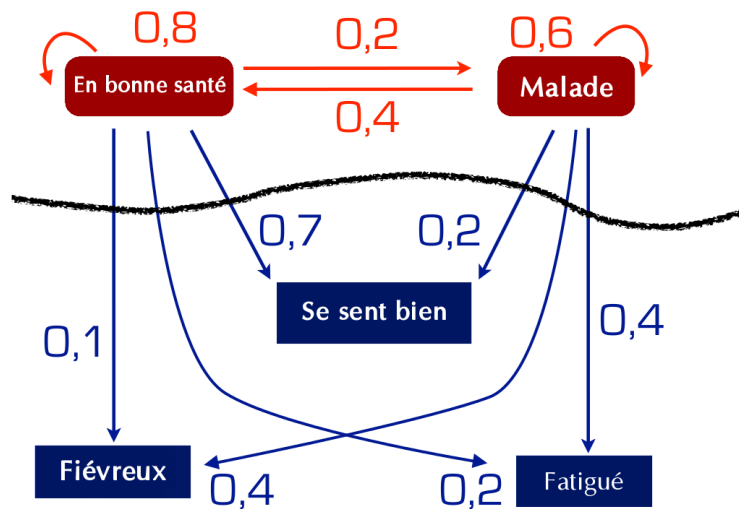


FIGURE 3.2 – Exemple d'un modèle de Markov caché

La spécificité des modèles de Markov cachés qui les rend si utiles en reconnaissance vocale est le fait qu'ils puissent apprendre et se perfectionner. En effet, il est possible de démontrer qu'en appliquant certaines formules sur l'automate à partir d'un mot (données par l'algorithme de Baum-Welch[3][13]), on parvient à l'améliorer et à le faire converger dans le domaine des automates vers un automate reconnaissant plus fidèlement le mot qu'on lui applique. Intuitivement, cela repose sur l'idée selon laquelle plus on "entraîne" l'automate à reconnaître un mot, plus il le reconnaîtra par la suite de manière fiable.

3.2.3 Modèles discrets et modèles continus

La figure qui vous a été présentée ci-dessus a un nombre fini de signaux, il s'agit de ce qu'on appelle un modèle de Markov caché *discret*. Il existe une version *continue* de ces automates où les signaux observés ne sont plus pris parmi un ensemble fini, mais forment en fait un continu d'observations. Un exemple pour illustrer ceci est le cas des couleurs. On peut soit considérer que l'ensemble des couleurs observées dans la vie de tous les jours forment un ensemble discret noir, rouge, vert, bleu, jaune, blanc etc..., ou envisager que cet ensemble est continu et qu'on le parcourt en balayant le spectre lumineux en passant par tous les dégradés. Les modèles de Markov cachés discrets et continus opposent deux conceptions similaires des observations effectuées. On comprend avec la métaphore des couleurs que considérer l'ensemble des observations comme discret alors qu'il est en réalité continu ne peut se faire sans perte drastique d'information. On serait en effet amené à assimiler les différents nuances de rouge bien qu'elles soient perçues différemment. C'est pourquoi, si l'on souhaite garder un maximum de fiabilité, il est nécessaire d'utiliser la version continue des modèles de Markov. Lorsqu'un état émet un signal, au lieu de chercher entre les différents signaux possibles, il effectue un calcul sur une combinaison

linéaires de fonctions gaussiennes dans l'espace à 12 dimensions, qui est la taille des MFCC¹. Les pics des gaussiennes représenteraient les signaux discrets. La nature des gaussiennes regroupe donc les probabilités autour de ces pics en conservant le caractère continu des fonctions

3.2.4 Application à la reconnaissance vocale

Les modèles de Markov cachés sont largement répandus dans la reconnaissance vocale[3][14][15]. Entre un modèle discret et un modèle continu, nous avons choisi ce dernier car les données en entrée ne font pas partie d'un ensemble fini : il existe une infinité de sons possibles pour un même phonème. Les modèles de Markov cachés sont particulièrement adaptés pour la reconnaissance vocale car ils permettent un apprentissage constant de la part du programme : celui-ci est capable d'apprendre de nouveaux mots de manière autonome, et de s'améliorer au-fur-et-à-mesure que la base de données de mots grandit.

Nous avons modélisé chaque mot par un automate, dont les états sont les différents phonèmes du mot. Lorsque l'on prononce un mot, on se dirige dans l'automate grâce aux phonèmes prononcés, jusqu'à rencontrer l'état final. Ceci permet de reconnaître le mot même si une syllabe dure plusieurs secondes : dans ce cas, on se contente de tourner en rond (en restant sur l'état 0 de la figure par exemple) dans l'automate jusqu'à rencontrer un nouveau phonème. Dans l'automate, la transition de l'état i à k représente la probabilité de passer de l'état i à k , c'est-à-dire la probabilité que le phonème n° k vienne tout de suite après le phonème n° i .

Exemple:

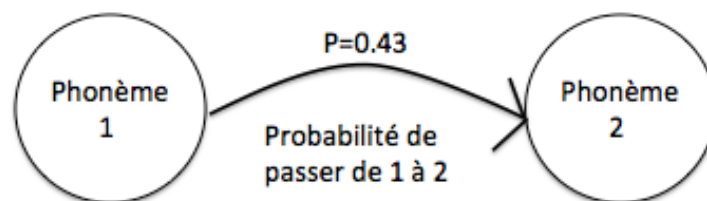


FIGURE 3.3 – Exemple de deux phonèmes et de la probabilité de passer du phonème 1 au phonème 2

3.3 Principaux algorithmes sur les modèles de Markov

Lorsque l'on fait passer un mot dans un automate, ie. qu'on s'oriente dans l'automate à l'aide des phonèmes, on peut calculer la probabilité que le mot corresponde à cet automate : on multiplie toutes les probabilités rencontrées pendant le parcours. Elles dépendent bien sûr du chemin parcouru (i-e des transitions rencontrées). C'est le principe de l'algorithme *forward*.

L'algorithme de *Baum-Welch* permet d'optimiser un automate. En se plaçant dans l'ensemble des modèles de Markov, on cherche à faire converger une suite d'automates définis à l'aide de plusieurs versions d'un même mot vers un automate optimisé qui corresponde au mieux au mot.

3.4 Application à notre objectif

Résumons la situation lorsque l'on lance notre programme : d'un côté une base de données de mots, représentés chacun par un automate; de l'autre, un fichier audio : le mot prononcé par l'utilisateur. Le programme se déplace dans chaque automate grâce au fichier audio, il s'oriente en fonction des phonèmes prononcés. Nous appellerons cette opération "faire passer un mot dans un automate".

1. Mel Frequency Cepstral Coefficients, cf. partie précédente

L'algorithme *forward* permet donc de calculer la probabilité qu'un automate corresponde au mot prononcé : en comparant les probabilités dans chacun des automates, on sélectionne la plus grande et on a l'automate qui correspond le mieux au mot sélectionné.

L'algorithme de Baum-Welch permet l'apprentissage de nouveaux mots : pour chaque nouveau mot il crée un nouvel automate, et le rend le plus optimisé possible en s'appuyant sur la bibliothèque existante. C'est ce que fait la partie logicielle de notre programme, pour que les programmeurs puissent agrandir la base de données.

3.5 Phase d'apprentissage

Une fois l'algorithme de reconnaissance vocale implémenté, il nous a fallu l'améliorer. Deux aspects demandent un apprentissage de la part du programme. Il doit d'abord faire grossir l'ensemble des mots reconnus, de manière à pouvoir en reconnaître le plus possible. Mais il est aussi intéressant de lui faire apprendre un mot par des locuteurs différents. Plus le nombre de locuteurs est grand, plus l'algorithme peut être précis.

Enregistrer plusieurs personnes permet d'obtenir une diversité de spectres qui accroît la précision du programme.

Une fois un mot appris, il est également très utile qu'un même locuteur enregistre de nombreuses versions du mot. Nous avons fait pour notre locuteur 10 versions de chaque mot.

Pour mettre en place un apprentissage, nous avons des besoins matériels (stocker l'ensemble des mots reconnus) mais aussi des besoins humains, et en l'occurrence une diversité de voix.

3.6 Phase de reconnaissance

La phase de reconnaissance constitue le cœur du programme. Comme dit précédemment, le programme effectue l'algorithme *forward* sur chacun des automates et renvoie le mot le plus probable, après avoir comparé toutes les probabilités.

A l'origine, la phase de reconnaissance a été codée en Python. Cependant le temps d'exécution étant trop long, nous l'avons donc codé en C++, ce qui a permis de diviser le temps d'exécution par 400. Grâce à ce travail laborieux, le programme s'effectue en un temps proche de la seconde. Tout a été mis en place, notamment en amont avec le codage en C++ de la transformée de Fourier rapide, pour privilégier la rapidité de l'exécution.

Au départ nous n'avions qu'un seul locuteur pour faire la base de donnée des mots reconnus, ce qui ne permettait de faire fonctionner le programme que pour un seul utilisateur : celui qui avait enregistré les mots. Cependant nous avons enregistré plusieurs locuteurs, ce qui permet au programme de reconnaître plusieurs utilisateurs, même un utilisateur qui n'aurait pas encore enregistré de mot.

3.7 Récapitulatif du principe de création du modèle de Markov caché

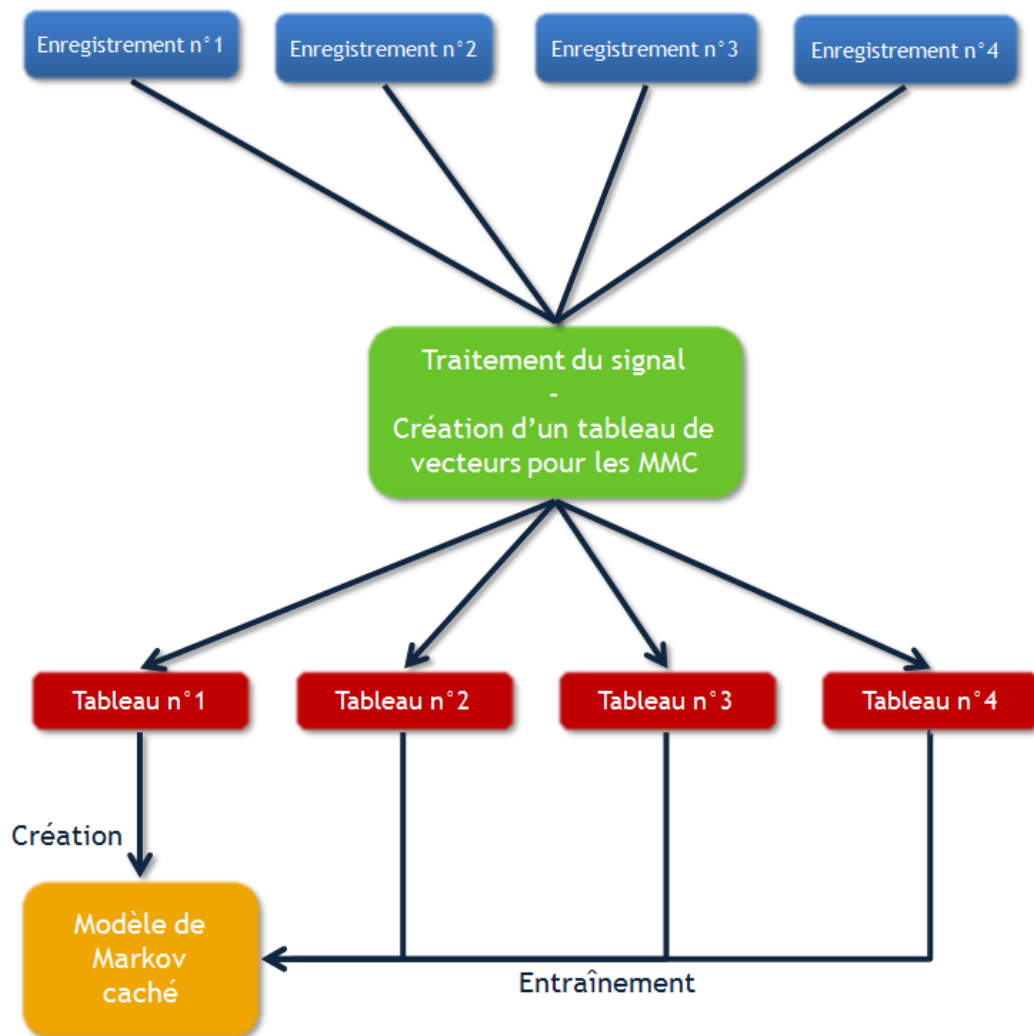


FIGURE 3.4 – Schéma récapitulatif du principe des modèles de Markov cachés

Récapitulatif

Voilà donc l'approche technique que nous avons choisie pour réaliser ce logiciel de reconnaissance vocale.

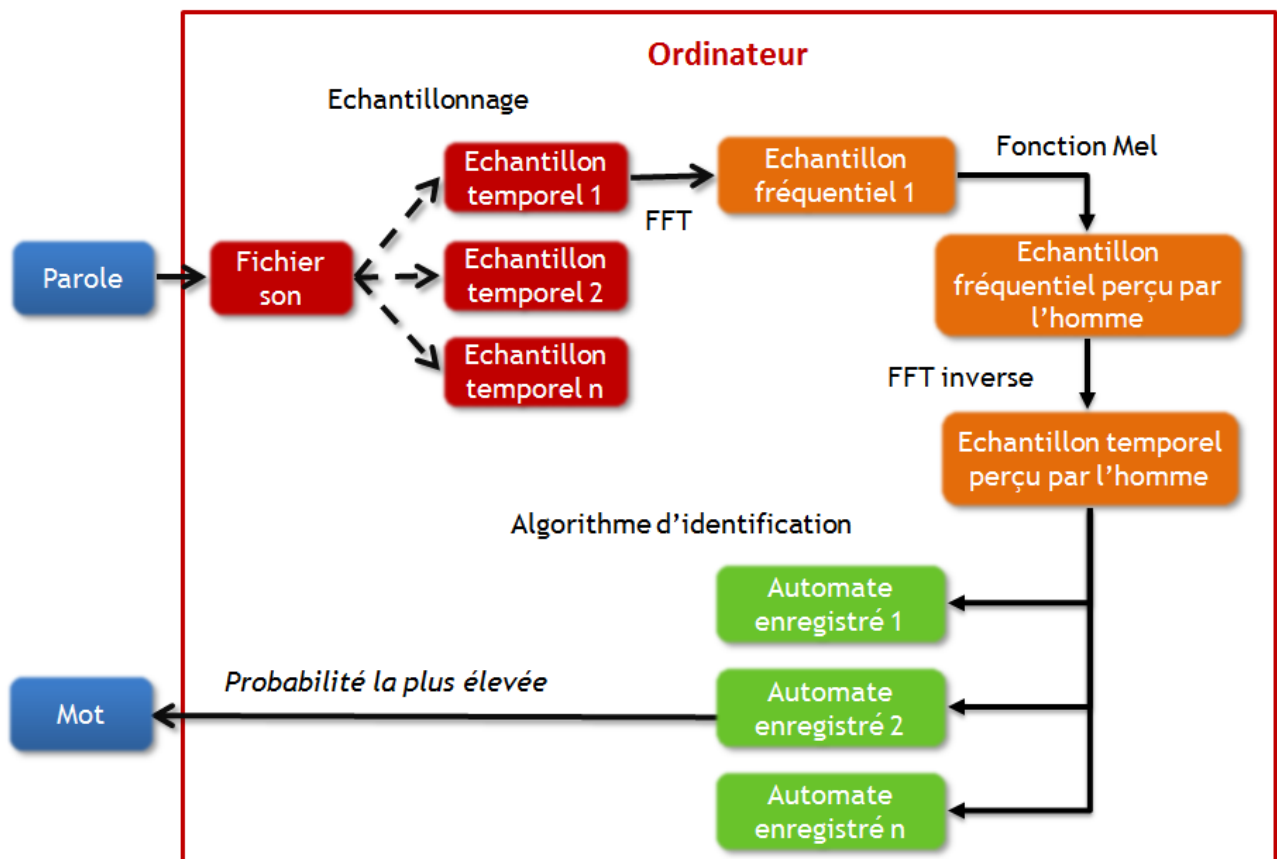


FIGURE 3.5 – Schéma du parcours global du son dans le programme et des différentes transformations qui lui sont appliquées

Les différentes étapes présentées plus haut consistent en l'algorithme codé. Pour cela, nous nous sommes répartis les tâches de manière à ce que chaque codeur ait en charge une partie de l'algorithme. Il a alors fallu, à la fin, mettre bout à bout les fragments d'algorithme pour les faire fonctionner, ce qui a là aussi demandé du temps.

Pour la répartition générale du code, nous avons confié la partie la plus sensible, les MMC, à deux codeurs. En effet, il n'était pas nécessaire d'être trop nombreux sur cette partie, cela aurait entraîné une baisse de productivité et certaines difficultés de communication. La partie de traitement du signal a réuni beaucoup plus de codeurs : 2 sur l'enregistrement et le recadrage ; 2 sur l'échantillonnage et le fenêtrage ; 1 sur la transformée de Fourier ; et 1 sur la transformée inverse. Enfin, deux autres élèves ont codé l'interface graphique dont nous parlerons à la partie suivante. Les élèves restants étaient chargés respectivement de rassembler les différents éléments en un programme unique, de concevoir l'application web et de procéder à l'étude économique.

Deuxième partie

Approche commerciale

1. Approche du développement du projet

Pour assurer une rentabilité à notre projet, il nous faut le penser, le structurer en vue d'une large distribution sous de multiples formes.

Nous sommes dotés d'une identité porteuse de ce projet. Le groupe de travail est baptisé The SpeechApp Company. Visuellement, elle se constitue en premier lieu d'un logo : Un micro, élément central du projet, dont la tête est le logo de Mines ParisTech, signe de notre appartenance à l'école et de l'aide qu'elle nous a apportée dans le projet.



FIGURE 1.1 – Logo de l'application

Les produits de The SpeechApp seront identifiables par un préfixe commun : Speech. Ils s'appelleront ainsi par exemple SpeechApp, SpeechServer ou SpeechRecorder.

1.0.1 Choix d'une architecture optimale pour notre projet

Distribuer notre projet tel quel présenterait à ce stade de nombreux défauts :

- le coeur de notre technologie de reconnaissance vocale est directement accessible à tous.
- une interface unique en ligne de commande constitue un blocage majeur pour la majorité des utilisateurs finaux et empêche une intégration large à des applications tierces.

Étudions l'opportunité d'adopter une architecture client/serveur pour ce projet.

Dans ce scénario, divers clients logiciels, potentiellement indépendants de The SpeechApp Company pourraient communiquer par requêtes/réponses (spécifiées par une API¹) avec les serveurs de The SpeechApp Company. Ces derniers seuls auraient accès au coeur algorithmique du projet, qui resterait

1. API : interface de programmation permettant d'utiliser les fonctions proposés par le serveur

ainsi exclusivement entre nos mains. Par leurs requêtes, les clients demanderaient l'analyse automatique de mots, l'ajout de nouveaux mots ainsi que toute autre opération pertinente relative à l'analyse et la gestion d'une base de données de mots. L'accès à notre API serait monétisable forfaitairement ou à l'utilisation.

Les mots enregistrés par les clients seraient conservés dans des bases de données chez The SpeechApp Company. La location de ces bases de données hébergées serait monétisable. Alors, The SpeechApp Company pourrait prioritairement développer deux applications connectables au serveur : la première, SpeechRecorder, permettrait l'enregistrement aisé de nouveaux mots dans les bases de données clients. La seconde, SpeechApp, permettrait, au travers d'une application Web riche, de tester la reconnaissance vocale en ligne.

Cette configuration permettrait aussi à une multitude d'applications tierces d'utiliser notre technologie en ne voyant de l'extérieur qu'une API définissant le format des requêtes et réponses dans la communication entre clients logiciel et serveur.

Nous aboutirions alors à l'architecture représentée par le schéma suivant :

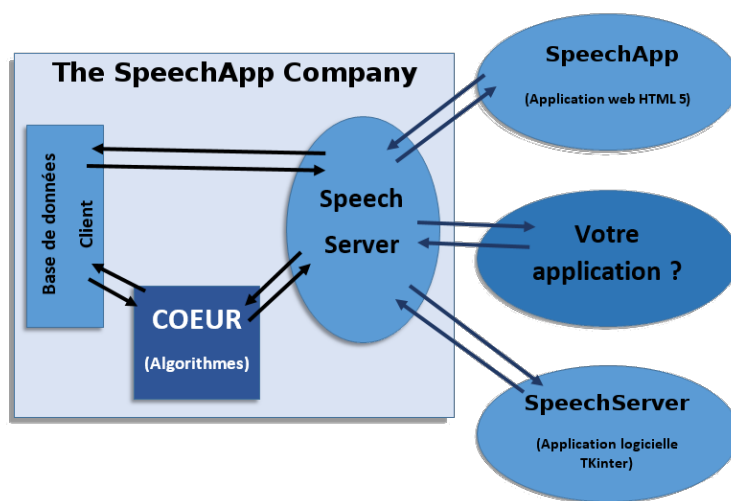


FIGURE 1.2 – Architecture proposée pour le projet The SpeechApp Company

Plus précisément dans le cadre des échanges entre le SpeechServer et les applications externes, les requêtes pourraient être traitées de la façon suivante : le client (au sens logiciel toujours) envoie au SpeechServer une requête HTTP ² POST ³ contenant un formulaire avec en particulier son identifiant, son mot de passe, la base de données qu'il veut utiliser, l'action qu'il veut faire effectuer au SpeechServer, et les données d'entrée qui lui sont associées. La requête analysée par le SpeechServer, les opérations adéquates ayant été réalisées par le coeur algorithmique, le SpeechServer répond au client par une réponse HTTP POST contenant des données au format XML ⁴. Le client peut alors lire et interpréter la réponse donnée par le SpeechServer.

Avec ces spécifications, nous obtiendrions le cycle suivant pour la reconnaissance d'un mot par SpeechApp :

2. HTTP est un protocole de communication client-serveur développé pour le web. (Wikipédia)

3. POST est une méthode spécifique de requêtes HTTP

4. XML est un langage informatique de balisage facilitant l'échange automatisé de contenus complexes (Wikipédia)



FIGURE 1.3 – Reconnaissance d’un mot par SpeechApp couplée au SpeechServer

L’architecture client/serveur proposée présenterait pour nous l’avantage de

- permettre la création d’un écosystème varié d’applications basées sur le coeur algorithmique de The SpeechApp Company via l’API de son SpeechServer, et générant ainsi des revenus
- conserver le coeur de notre travail entre nos mains et même de nous donner le contrôle sur toute la chaîne

L’architecture client/serveur proposée présenterait pour nos clients l’avantage de

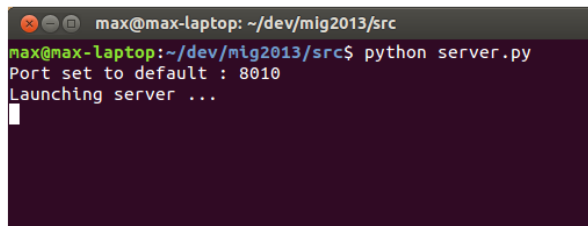
- ne pas se soucier du coeur algorithmique de la reconnaissance vocale, en n’y voyant que l’API de SpeechServer. Cette API peut offrir par ailleurs une grande liberté d’action
- n’avoir pas ou peu d’investissement initial de développement à effectuer, nos applications propriétaires SpeechApp et SpeechRecorder pouvant être intégrées sous forme de widgets aux applications tierces
- ne pas avoir à faire de lourds calculs eux-mêmes, ceux-ci étant réalisés par les machines de The SpeechApp Company
- les cycles de mises à jour seraient en majorité invisibles chez les clients, l’API restant immuables sur des cycles plus long (Long Term Support)

Au vu des nombreux avantages qu’elle présente, *nous avons donc opté pour une architecture modulaire client/serveur pour notre projet.*

1.0.2 Réalisation du SpeechServer

Le SpeechServer a été codé en Python. Python a une librairie standard suffisamment riche pour n’avoir à traiter ce problème qu’à un haut niveau (en réception de requêtes selon leurs méthodes). De plus, ce choix facilite les interactions avec le coeur algorithmique : des imports et appels de fonctions depuis le SpeechServer suffisent.

Finalement, le SpeechServer prend seulement la forme d’un programme Python à lancer sur un ordinateur.



```
max@max-laptop: ~/dev/mig2013/src
max@max-laptop:~/dev/mig2013/src$ python server.py
Port set to default : 8010
Launching server ...
```

FIGURE 1.4 – Le SpeechServer lancé

Il écoute alors les requêtes sur le port 8010 (par défaut) de l'ordinateur. Lorsqu'il en reçoit, il interagit avec le coeur algorithmique et le système de gestion de bases de données mis en place.

1.0.3 Système de Gestion de Base de Données (SGBD)

Le SGBD doit permettre de stocker et gérer les fichiers audio associés aux mots (au moins une dizaine d'enregistrements par mot), les modèles de markov cachés qui leur sont associés ainsi que les données d'authentification des applications clientes.

Le standard actuel de gestion de bases de données est le modèle relationnel basé sur le langage SQL. Néanmoins, dans le cas précis de stockage de fichiers relativement lourds (> 0.1 Mo), la lecture/écriture des données directement sur le disque dur s'avère plus performante.

Nous avons donc fait le choix de stocker nos données sur le disque dur du serveur, en enregistrant les fichiers audio en format brut, et les autres données (modèles de Markov, données d'authentification) comme des objets Python, avec le module pickle de la librairie standard.

Un module python db.py a été développé par nos soins pour gérer efficacement nos fichiers

Comme les accès en lecture/écriture à la mémoire RAM sont bien plus rapides que les accès aux disques durs, *on pourrait obtenir un gain de vitesse significatif pour la reconnaissance vocale en chargeant l'intégralité des données en mémoire RAM au démarrage du SpeechServer*. La vitesse en lecture/écriture sur un disque dur est de l'ordre de 50 Mo / s. Sur la RAM, elle est de l'ordre de 1 Go / s, soit un gain d'un facteur 20 pour les opérations en mémoire.

Néanmoins, la quantité de mémoire RAM nécessaire serait très importante, croissant linéairement avec le nombre de mots enregistrés. Les coûts engendrés pourraient être importants.

Tâchons de dimensionner l'infrastructure serveur dont nous aurions besoin.

1.1 Dimensionnement de l'infrastructure de calcul de The Speech App Company

Il nous faut d'abord définir les variables relatives au fonctionnement commercial de The Speech App Company ainsi que leurs valeurs de référence.

1.1.1 Hypothèses de fonctionnement

Soit M le nombre de clients de The SpeechApp Company. La référence sera $M = 1000$. Soit N le nombre moyen de mots dans les bases de données de chaque client. Nous prenons pour référence $N = 2000$ mots : un dictionnaire comme le Petit Robert en contient 60000.

Soit J le nombre de requêtes par seconde. Nous prendrons pour référence $J = 1000$ requêtes / s

On vise le traitement des requêtes en 1s. On gère donc J requêtes en simultané.

Les fichiers audio bruts envoyés par les clients au serveur pèsent environ 100 ko chacun. Les Modèles de Markov Cachés (MMC) associés aux mots pèsent environ 50 ko pour chaque mot. Lors des traitements sur ces fichiers audios, on estime qu'on a besoin de créer 5 fichiers audios temporaires, d'environ 100 ko chacun.

1.1.2 Dimensionnement en mémoire RAM et espace disque

Les fichiers audios bruts (10 par mot par défaut) et les MMC sont conservés sur le disque dur. Il faut donc $(10 * 100ko + 50ko) * N * M = 2.100To$ d'espace sur le disque dur.

Par ailleurs, on charge les MMC en mémoire, soit un espace RAM nécessaire de $50ko * N * M = 100Go$

Lors des opérations, si les $5 * 100 ko$ de fichiers temporaires sont créés en RAM, et qu'on gère environ $J = 1000$ requêtes en parallèle, il nous faut 500 Mo de RAM en plus, ce qui est marginal.

Au vu des capacités mémoire en informatique, toutes puissances de 2, *Dans le cadre de référence, il nous faut au moins 128 Go de RAM et 4 To d'espace disque*

1.1.3 Dimensionnement réseau

Pour la reconnaissance de mots, tâche la plus courante, Le serveur reçoit J requêtes de 100 ko (fichiers audio). Il faut donc recevoir 100 Mo / s de données. Le débit descendant (vers le serveur) doit donc être supérieur strictement à 100 Mo / s. Le serveur répond par des fichiers ne contenant que du texte, de taille négligeable devant celle des fichiers audio. Le débit montant (depuis le serveur) n'est donc pas un facteur discriminant dans le choix d'une connexion au réseau.

On veillera à avoir une connexion d'au moins 200 Mo / s)

1.1.4 Dimensionnement des éléments de calculs

La puissance de calcul d'un ordinateur se mesure communément en Flops, acronyme désignant le nombre d'opérations en virgule flottante que peut effectuer un ordinateur. Aujourd'hui, les ordinateurs de bureau ont une puissance de l'ordre du Giga-Flops tandis que les calculateurs les plus performants dépassent la dizaine de Peta-Flops.

Sur un ordinateur d'une puissance de calcul de 1 GFlops, on observe que lors de la reconnaissance d'un mot, l'unique opération dont la complexité dépend du nombre de mots en jeu, l'exécution de l'algorithme Forward (linéaire) prenait 0.005s sur une base de 100 mots.

Ainsi pour réaliser $J = 1000$ reconnaissances en simultané sur des bases de $N = 1000$ mots avec un temps d'exécution de l'algorithme Forward de moins de 0.5s, *il nous faut une puissance de calcul d'au moins 100 Gflops.*

Les processeurs de dernière génération dédiés au calcul atteignent ce niveau de performance. Le Intel Xeon E5-2670 atteint ainsi en théorie 330 Gflops

1.1.5 Choix de l'infrastructure et coûts induits

Connaissant les caractéristiques minimales du serveur : en termes d'espace RAM, disque dur, de connexion réseau et de puissance de calcul, nous pouvons choisir le serveur le plus adapté à nos besoins.

L'hébergeur OVH propose une gamme de serveurs de calcul pour les entreprises :

GAMME ENTERPRISE 2014				
Modèle	SP-64	SP-128	MG-128	MG-256
Prix	81.99€ HT /Mois	131.99€ HT /Mois	202.99€ HT /Mois	302.99€ HT /Mois
Installation	99.99€ HT	99.99€ HT	99.99€ HT	99.99€ HT
CPU	Intel Xeon E5-1620v2	Intel Xeon E5-1650v2	Intel Xeon 2x E5-2650v2	Intel Xeon 2x E5-2670v2
Cores / Threads	4c / 8t	6c / 12t	16c / 32t	20c / 40t
Fréquence / Burst	3.7 GHz+ / 3.9 GHz+	3.5 GHz+ / 3.9 GHz+	2.6 GHz+ / 3.4 GHz+	2.5 GHz+ / 3.3 GHz+
RAM	64 Go DDR3 ECC	128 Go DDR3 ECC	128 Go DDR3 ECC	256 Go DDR3 ECC
Disques Durs	2x 2 To SATA3 SSD ⁽¹⁾ / SAS ⁽¹⁾	2x 2 To SATA3 SSD ⁽¹⁾ / SAS ⁽¹⁾	2x 2 To SATA3 SSD ⁽¹⁾ / SAS ⁽¹⁾	2x 2 To SATA3 SSD ⁽¹⁾ / SAS ⁽¹⁾
RAID	SOFT HARD ⁽¹⁾	SOFT HARD ⁽¹⁾	SOFT HARD ⁽¹⁾	SOFT HARD ⁽¹⁾
Bande passante	300 Mbps ⁽¹⁾	300 Mbps ⁽¹⁾	400 Mbps ⁽¹⁾	500 Mbps ⁽¹⁾

FIGURE 1.5 – Gamme de serveur de calculs d'OVH

Nous devons disposer de 128 Go de RAM, de 4 To d'espace disque, de 100 Mo/s de connexion au réseau. Aussi le processeur Intel Xeon E5-1650 v2 ne dépasse pas les 70 Gflops et ne valide donc pas le critère de puissance de calcul. Le bi-ES-2650 atteint 140 Gflops, et le bi-ES-2670 330 Gflops.

Afin d'avoir une certaine marge, nous préférons prendre le processeur bi-Intel Xeon ES-2670. Nous sélectionnons donc le serveur MG-256 de OVH pour 303 euros HT / mois qui valide nos critères de performance avec une marge conséquente. À des fins de redondance, nous aurions besoin de 2 serveurs identiques, pour donc 606 euros HT / mois.

Nous nous sommes contentés d'un stockage des données intégralement sur disque dur et de moyens bien plus réduits lors de la phase de développement.

Les spécifications du SpeechServer et du SGBD ayant été définies, il devient possible et nécessaire de construire des applications se fondant dessus.

1.1.6 SpeechRecorder

Il est nécessaire de proposer aux clients une interface plus simple à appréhender que la console. C'est pourquoi nous avons développé l'application logiciel SpeechRecorder, qui permet aux clients enregistrés dans nos bases de données d'authentification (s'acquittant d'une licence), d'ajouter des mots à leurs bases de données. Elle devrait permettre à terme, de gérer l'intégralité des bases de données client.

Cette interface a été réalisée avec la librairie TKinter de Python, la librairie graphique Python la plus simple et la plus largement disponible : elle est incluse dans les paquetages de base de Python.

FIGURE 1.6 – Authentification d’un client au SpeechRecorder

FIGURE 1.7 – L’interface du SpeechRecorder

Pour entrer un nouveau mot dans une base de données client, par défaut 10 enregistrements sonores sont pris par SpeechRecorder. La librairie additionnelle pyaudio est utilisée pour ce faire.

1.1.7 SpeechApp

SpeechApp est le démonstrateur principal de notre projet. Il s’agit d’une application web permettant au grand public de tester notre technologie de reconnaissance vocale de mots isolés. À l’aide des dernières APIs HTML5 (élaborées depuis le début d’année 2013), l’utilisateur peut s’enregistrer sans l’installation de logiciel auxiliaire. Son enregistrement audio est transmis au SpeechServer (selon le schéma spécifié plus haut) qui renvoie le mot trouvé.

Pour ce démonstrateur, une application web a été choisie car elle fonctionne sur tout terminal doté d’un navigateur web récent sans nécessiter la moindre installation : nous l’avons conçue de façon à ce qu’elle soit adaptée aussi bien aux grands écrans d’ordinateurs, qu’à ceux plus petits des tablettes et smartphones. On qualifie ce type de design de "Responsive".



FIGURE 1.8 – Le démonstrateur SpeechApp sur ordinateur



FIGURE 1.9 – Le démonstrateur SpeechApp sur iPhone

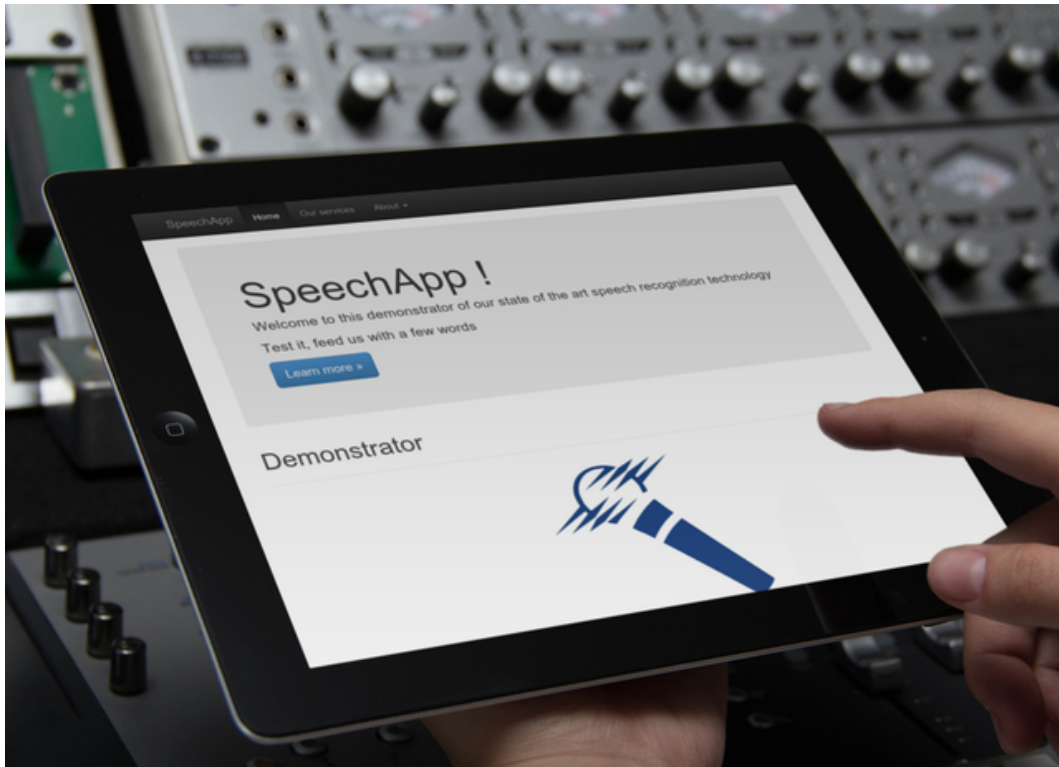


FIGURE 1.10 – Le démonstrateur SpeechApp sur iPad

Une difficulté néanmoins a été de rendre l'enregistrement audio fonctionnel sur la majorité des navigateurs. Nous assurons la compatibilité pour les versions récentes des moteurs de rendu Gecko et WebKit soit les dernières versions de Firefox, Chrome et Safari ainsi que leurs éditions mobiles.

SpeechApp n'a en elle-même pas d'autre fonction que celle de démonstrateur, néanmoins ses modules peuvent être distribués aisément, sous forme de widgets intégrables n'importe où.

De plus, une application web reprenant des modules de SpeechApp pourrait être transformée aisément en application native pour smartphone iOS, Android, Windows Phone, Firefox Mobile ou encore en application Windows 8. Des frameworks open-source font ce travail presque automatiquement (par exemple <http://phonegap.com/>).

2. Étude de marché et applications

2.1 Le marché actuel

Bien qu'aujourd'hui restreint, le marché de la reconnaissance vocale est appelé à grandir dans les prochaines années. Si les systèmes de reconnaissance vocale sont fréquents pour les objets multimédias à usage personnel (comme les ordinateurs portables ou les téléphones mobiles), ils ne servent qu'à simplifier certaines tâches de l'utilisateur, et leurs performances moyennes n'incitent pas à s'en servir davantage. Pour les téléphones mobiles, le leader est **Siri**, mais il reste peu utilisé pour les raisons ci-dessus.

Dans le domaine des logiciels payants, pour un usage plus sérieux, le marché est dominé par les logiciels **Dragon NaturallySpeaking**[16] de la firme américaine Nuance. Les prix varient entre 100\$ pour le modèle de base et 1000\$ pour les versions professionnelles. Plus la base de données de mots est grande, plus les erreurs sont fréquentes. C'est pourquoi Dragon NaturallySpeaking[16] propose des versions adaptées à un domaine particulier. Il en existe par exemple une dédiée aux métiers juridiques, avec une base de données composée d'un vocabulaire technique de droit, et une autre dédiée au domaine de la santé et contenant des termes médicaux. Ces versions visant un marché plus restreint, leurs prix est plus élevé. Cependant, la demande étant en constante augmentation (un tiers des radiologues français utilisent cette technologie, tout comme 95% des hôpitaux aux Pays Bas), le marché est assez prometteur. En effet, cette technologie réduit considérablement les tâches administratives de ces professions : une simple relecture est suffisante. Cette compétitivité est renforcée par des taux de réussite exceptionnels grâce à des bases de données adaptées, et par l'absence de concurrence forte.



FIGURE 2.1 – Entreprises actuellement sur le marché de la reconnaissance vocale

Dans notre cas, nous avons choisi de concevoir un logiciel avec une base de données dédiée essentiellement à un usage personnel. En effet, dans le temps qui nous était imparti, créer des bases de données spécialisées nous paraissait très compliqué. Il aurait fallu faire une étude linguistique très poussée pour construire la base de données, alors que nous avons concentré l'essentiel de nos efforts sur l'algorithme de reconnaissance lui-même. Notre produit est donc destiné à un usage plus ludique, ou du moins personnel. Notre cible est donc légèrement différente, puisque les professionnels intéressés par notre produit doivent construire leurs propres base de données. Cette approche a le désavantage d'être parfois fastidieuse pour l'utilisateur, celui-ci devant ajouter soi-même les mots. Néanmoins, la reconnaissance sera plus fiable puisque la voix de l'utilisateur servant elle même de comparateur, elle permet d'avoir une base de données réellement personnalisée (celles de Dragon, bien que spécialisées,

ne sont pas personnelles). Le prix envisagé de la licence pour cette utilisation du logiciel est comparable à celui des versions personnalisées de Dragon (de l'ordre de 1000 €). La version à usage personnel, sans possibilité d'ajouts de mots, sera elle au prix de 5€, bien qu'il soit difficile de prévoir son potentiel, les concurrents étant très nombreux, et les rapports qualité-prix très variables.

2.2 Principaux domaines d'application

La reconnaissance vocale est une technologie promise à un futur radieux. Les plus grands noms de l'informatique, dont Bill Gates¹, annonçaient ainsi il y a quelques années qu'elle allait remplacer les claviers d'ici peu. Et bien qu'il s'avère aujourd'hui que leurs prédictions ne se soient pas encore réalisées, il paraît probable qu'elles se concrétisent plus tard. En effet, le principal obstacle à l'explosion de cette technologie étant le manque de fiabilité total, le temps et les progrès qui l'accompagnent devraient rendre la technologie de plus en plus sûre.

L'armée états-unienne a bien compris le potentiel de cette technologie et investit massivement depuis des années pour la développer[17][18]. Elle est d'ailleurs déjà utilisée sur certains avions de chasse et hélicoptères aux Etats-Unis, mais également en France, au Royaume-Uni et en Suède. Vu les investissements massifs, il y a matière à penser que les armées de ces différents pays ont des techniques bien plus avancées que celles connues du grand public, déjà plutôt performantes. Pour le moment, les commandes vocales ne servent pas encore à des fonctions critiques comme lancer un missile, et demandent toujours la confirmation du pilote avant d'exécuter une action. Elles libèrent néanmoins considérablement le pilote de beaucoup de tâches secondaires, ce qui lui permet de se concentrer sur les fonctions critiques. Elles demandent néanmoins une grande fiabilité dans des conditions de stress et de bruit ambiant énorme (en particulier pour les hélicoptères, dans lesquels les pilotes n'ont souvent pas de casque anti bruit). Dans ce domaine, les perspectives sont donc très intéressantes financièrement mais requièrent un savoir-faire qui semble hors de notre portée.

La reconnaissance vocale est également utilisée dans le contrôle aérien[19], et pourrait à terme remplacer les contrôleurs aériens. En effet, les phrases utilisées dans ce contexte sont très typées, ce qui favorise la reconnaissance (phrases souvent identiques, syntaxe très simple, prononciation très articulée). La technologie est donc moins avancée que dans le domaine de l'armée mais est déjà utilisée aux Etats-Unis, en Australie, en Italie, au Brésil et au Canada. Notre produit pourrait servir à ce type d'application, en créant une base de données spécifique au contrôle aérien.

Comme vu précédemment, la reconnaissance vocale se développe dans de nombreux domaines professionnels où les tâches administratives prennent beaucoup de temps, notamment la médecine[20][21], le droit et la police. Elle est notamment déjà utilisée dans 95% des hôpitaux aux Pays-Bas pour faciliter la rédaction de compte-rendus. Pour le droit, elle pourrait remplacer le travail du greffier pour prendre des notes dans les tribunaux. Et pour la police[22], elle permet de rédiger des rapports environ trois fois plus vite qu'au clavier. Le besoin de fiabilité est bien moindre dans ces domaines que dans les domaines de l'armée ou du contrôle aérien, une relecture étant souvent largement suffisante. Dans le domaine du droit, il faut néanmoins prendre en compte les conditions particulières d'enregistrement (brouhaha ambiant, émotions dans la voix, volume variable...).

Une autre application possible de la reconnaissance vocale est l'aide aux handicapés[23], par exemple pour contrôler une chaise roulante. Les phrases utilisées étant très typées (avancer, reculer,...), la technologie n'a pas besoin d'être très avancée. De plus, avec la possibilité qu'offre notre produit d'ajouter ses propres mots à la base de données, l'utilisateur lui-même pourrait rentrer les commandes et donc avoir un taux de reconnaissance très élevé.

1. <http://www.clubic.com/article-161030-3-clubic-test-solutions-reconnaissance-vocale.html>



FIGURE 2.2 – Micro QuadMouse qui peut être utilisé exclusivement avec le menton, les lèvres et la langue. Il possède un logiciel de reconnaissance vocale intégré

Cette technologie est également très utilisée pour un usage plus ludique : fonctions de recherche sur les téléphones mobiles ou ordinateurs, robotique, jeux vidéo, traduction automatique,...

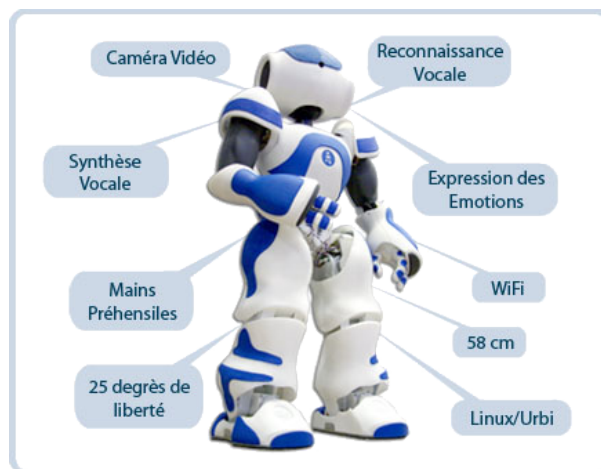


FIGURE 2.3 – Nao, un robot possédant un logiciel de reconnaissance vocale.

Notre produit, dans sa version pour particuliers, peut servir à ces usages, mais comme nous le verrons plus loin, la concurrence dans ces domaines est nettement plus forte que dans les domaines professionnels.

Enfin, la reconnaissance vocale peut servir à des fins sécuritaires, pour des vérifications d'identité. Il s'agit alors de reconnaître le locuteur, ce que notre produit ne permet pas.

Pour conclure, les applications de notre produit sont très nombreuses, et la demande est de plus en plus forte, ce qui montre sa pertinence économiquement parlant ; la qualité du produit est bien sûr néanmoins une condition indispensable à son succès.

3. Budget, modèle économique

3.1 Introduction

Après les études techniques et théoriques, l'étude économique est une nécessité. Elle est au coeur des problématiques de l'ingénieur, car c'est elle qui permet de dire si le projet est viable ou non.

Dans le cas de la programmation d'un logiciel de reconnaissance vocale, divers facteurs sont à prendre en compte, comme les salaires des employés, la communication sur le produit ou les impôts à payer. Il s'agit également de trouver le meilleur moyen pour vendre le logiciel. Faut-il le vendre pour iPhone sur l'App Store ? Le réserver à un public restreint (majoritairement des entreprises) ou le proposer également à des particuliers ?

La concurrence importante nous oblige à être à la fois ambitieux et prudent. Nous avons donc décidé d'envisager à la fois la vente sur notre site internet d'un logiciel pour les particuliers, et de proposer des licences en parallèle, permettant notamment aux entreprises d'accéder à nos bases de données, les compléter et créer leurs propres dictionnaires.

L'étude suivante tente de calculer d'abord le coût des salaires, pour parvenir à dresser le compte de résultat prévisionnel, préalable au bilan final et aux impôts qui vont suivre.

3.2 Les salaires

Treize employés travaillent sur le projet, pendant un temps effectif d'environ un mois. Parmi eux, un chargé des ressources humaines pour un salaire de 2750 € brut mensuel[24], un chargé d'étude de marchés, pour un salaire de 2700 € brut mensuel, les autres étant considérées comme des développeurs de moins de deux ans d'expérience, avec un salaire de 2290 € brut mensuel [25].

Sur ce salaire brut, l'employé paye environ 22% de charges salariales, et l'entreprise 44% de charges patronales. A l'aide de ces chiffres salariaux, on peut dresser le compte de résultat prévisionnel ci-dessous.

SALAIRES

Catégorie	Salaire brut	Charges salariales	Salaire net	Charges patronales	Budget
Personnel					
David Vitoux	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Axel Goering	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Sofiane Mahiou	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Maxime Ernoult	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Adrien De La Vaissière	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Clément Joudet	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Clément Roig	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Anis Khlif	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Paul Mustière	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Matthieu Denoux	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Julien Caillard	2290,00 €	503,80 €	1786,20 €	1007,60 €	3 297,60 €
Nathanaël Kasriel	2750,00 €	605,00 €	2145,00 €	1210,00 €	3 960,00 €
Thomas Debarre	2750,00 €	605,00 €	2145,00 €	1210,00 €	3 960,00 €
Total	30690,00 €	6751,80 €	23938,20 €	13503,60 €	44193,60 €

FIGURE 3.1 – Salaires

3.3 Le compte de résultat prévisionnel

Le compte de résultat prévisionnel dresse l'ensemble des charges (fixes et variables) de l'entreprise, ainsi que ses produits (recettes). Il permet de dresser ensuite le bilan prévisionnel, qui viendra dans la partie suivante.

On va montrer dans la suite de l'étude que, pour parvenir à un équilibre budgétaire, il nous faut, pour la première année, vendre 26000 logiciels à un prix de 4,17 € hors taxes, et une dizaine de licences permettant d'accéder à nos bases de données pour un prix de 833,33 €. Au niveau des charges, l'ensemble des salaires cités plus haut est à prendre en compte, ainsi que le coût de notre campagne de publicité. Celle-ci peut être décrite en deux principaux pôles : des articles de journaux spécialisés, gratuits, et des annonces google. On peut estimer le prix d'une telle annonce à 10 centimes d'€ le clic. En estimant que 10% des visiteurs du site par l'intermédiaire de l'annonce vont acheter le produit, on peut évaluer le coût de la publicité à 26 000 €. Enfin, l'entreprise aura besoin, pour parvenir à fournir ses services de deux serveurs capable de traiter 1000 requêtes par seconde sur une base d'un million de

mots pour un total de 606 € par mois.

La différence des produits et des charges donne alors un chiffre de 62 710 €.

COMPTE DE RÉSULTAT PRÉVISIONNEL

	Produit					Charges	
	Vente	Prix unité (Hors taxes)	Prix unité TTC	Nombre	Total	Salaires	45 777,00 €
	Logiciel	4,17 €	5,00 €	24000	120 096,00 €	Frais pub (google)	24 000,00 €
	Licences	833,33 €	1 000,00 €	10	10 000,00 €		
Total	60319,00 €						

FIGURE 3.2 – Compte de résultat prévisionnel

3.4 Le bilan

Actifs		Passif	
Actifs incorporels	0,00 €	Fonds propres	0,00 €
Créances	0,00 €	Dettes long terme	100000,00 €
Actifs immobiliers	0,00 €	Compte de Résultat prévisionnel	60319,00 €
Créances clients	0,00 €		
Trésorerie	0,00 €		

FIGURE 3.3 – Bilan

Le bilan prend en compte l'actif et le passif de l'entreprise. C'est grâce à lui qu'on peut calculer les bénéfices, et l'impôt sur les société qui suivra.

Cette année, celle-ci n'a pas d'actif réel. Pas de trésorerie, de créances ou d'actifs immobiliers et incorporels. Son passif ne contient pas de fonds propres, et le compte de résultat prévisionnel a été explicité plus haut. On peut en revanche considérer que nous avons effectué un prêt à long terme de 100 000 €, afin de financer les primes du projet.

3.5 Les impôts

S'agissant des impôts, nous devons dans un premier temps reverser à l'Etat la TVA sur les produits que nous vendons, à un taux de 20% à compter du 1er Janvier 2014. Le logiciel étant vendu 4,17 € et la licence 833,33 €, le total de la TVA à reverser sera de 27 020 €.

Ensuite, l'impôt sur les sociétés est à un taux de 33% sur les bénéfices. A partir du bilan et de la TVA, on peut estimer nos bénéfices à 34 690 €, et donc un impôt sur les bénéfices à hauteur de 11 448 €.[26]

Impôts		
Sur les sociétés	33% des bénéfices	11318,93 €
TVA	20% sur les ventes	26 019,20 €

FIGURE 3.4 – Impôts

3.6 Conclusion et vue sur le long terme

Grâce à l'étude précédente, et en considérant un prêt à un taux de 3% sur cinq ans, et le prêt de locaux et ordinateurs par un incubateur (l'école des Mines par exemple) l'entreprise est viable la première année à partir de 25 000 téléchargements. On aurait en effet un bénéfice net d'impôt supérieur aux 23 000 € nécessaires au remboursement.

En utilisant les mêmes calculs, pour les quatre années qui suivent, sans faire de mise à jour, il faudrait en moyenne 13 000 ventes de logiciels par an, et 3 ventes de licences.

Conclusion

Notre équipe de 13 étudiants en première année au sein de MINES Paristech s'est attachée pendant trois semaines à la conception complète d'un logiciel de reconnaissance automatisée de la parole. La reconnaissance vocale est l'un des domaines d'avenir de l'informatique et ses applications actuelles sont déjà nombreuses et variées (elles concernent entre autres domotique, défense ou encore dictionnaires en lignes).

Bien que la conception elle-même des algorithmes en Python et C++ (reconnaissance même des mots mais aussi traitement du signal en amont) fut le pilier de notre projet, une attention particulière a été portée aux perspectives économiques, le but étant de produire un produit pleinement commercialisable. Ce fut l'occasion de mettre à l'épreuve notre capacité à travailler en équipe : en effet, il a été nécessaire à la fois d'identifier et de répartir les tâches à effectuer, tout en s'assurant de la cohérence de tous nos pans de codes. Nous n'aurions pu y parvenir sans une bonne coordination, ni sans utiliser la plateforme Github, qui a permis un travail simultané et constructif de plusieurs codeurs sur un même sous-programme.

Notre produit final, nommé *SpeechApp*, est une plateforme logicielle et web complète offrant deux services distincts : la création d'une base de données de mots à partir d'enregistrements audios successifs et l'identification d'un enregistrement par rapport à une base de données de mots. L'accent a été mis sur l'ergonomie : l'utilisateur peut en seulement quelques clics reconnaître ou enregistrer un mot. De plus, notre interface est transportable sur la grande majorité des supports informatiques (ordinateurs, smartphones, tablettes) et est peu consommatrice en ressources. En effet, nos algorithmes étant opérés sur un serveur physique dédié, l'utilisateur n'a besoin que d'une faible capacité de calcul.

Derrière cette interface très facile d'accès se cachent cependant à la fois une théorie mathématique complexe (la théorie des modèles de Markov cachés) et une importante réflexion sur le traitement du signal sonore, de façon à se rapprocher du comportement de l'oreille humaine. Le résultat est très encourageant : nous avons pu créer notre propre « dictionnaire » audio, dont chacun des mots est bien reconnu par le logiciel. Nous envisageons cependant une optimisation de nos algorithmes de façon à identifier entre autres le sexe de l'auteur, son niveau de stress et la langue parlée. En seulement trois semaines de travail, nous sommes parvenus à construire les briques logicielles nécessaires au traitement du signal, aux calculs relatifs aux modèles de Markov cachés et à les rassembler. L'infrastructure du SpeechServer a été assise sur le cœur algorithmique, et ses applications clientes, SpeechRecorder et le démonstrateur SpeechApp ont été bâties avec succès.

L'étude économique que nous avons réalisée montre que notre projet est viable : cette affirmation, certes hardie, s'appuie sur l'analyse de notre valeur ajoutée comparée aux autres versions gratuites (par exemple Siri) ou payantes (Dragon) : nous offrons à l'utilisateur la possibilité de créer sa propre base de donnée spécialisée (ce qui implique qu'il pourra lui même la mettre en vente). De plus, tous nos développements ont été structurés de façon à faciliter notre développement commercial. Par conséquent, nous estimons que si notre produit venait à être commercialisé, il serait à même de concurrencer les produits déjà existants, ou du moins de susciter l'intérêt de leurs concepteurs en vue d'un éventuel rachat.

Plus largement, nous espérons que le travail effectué pour le développement du projet SpeechApp aura une utilité plus large que celle du MIG, c'est pourquoi en l'absence de lancement commercial de notre part, nous laisserons les sources du projet, que nous estimons être de qualité et ce rapport librement accessibles afin qu'ils puissent plus tard servir à autrui. N'oublions surtout pas que l'aventure de la réalisation d'une ébauche de The SpeechApp Company nous a permis à tous d'acquérir une culture minimale sur le développement d'un projet informatique, la gestion d'un groupe de travail de 13 personnes sans hiérarchie ainsi que sur la mise au point d'un modèle de développement commercial.

Troisième partie

Code

A. Code Principal

A.1 shell.py

```
1 def main(verbose=True, action=-1, verboseUltime=True):
2     db = Db("../db/", verbose=verbose)
3     #HMMs = {}
4     #db.addFile("hmmList.txt", HMMs)
5     choice = -1
6     while(not choice in range(1,8) ):
7         try:
8             if verboseUltime:
9                 choice = int(input("Que voulez-vous faire ?\n1-Enregistrer un element
10                 \n\
11 2-Realiser l'analyse d'un mot\n3-Tester\n4-Afficher resultats intermediaires\n5-
12  Gestion des fichiers de la base de donnees\n\
13 6-Creation d'un HMM\n7-Gerer les HMM\n-----\n"))
14         else:
15             choice = 2
16         except NameError:
17             print "Ceci n'est pas un nombre !"
18             #####
19             ###      ENREGISTREMENT      ###
20             #####
21             if choice == 1:
22                 #Realiser un enregistrement
23                 recorder(db)
24
25             #####
26             ###      ANALYSE D'UN SON      ###
27             #####
28             elif choice == 2:
29                 fileOk = False
30                 while not fileOk:
31                     #On choisit le dossier a afficher
32                     print "Voici la liste des mots a etudier : "
33                     dirList = db.printDirFiles("waves/")
34                     dirChoice = -1
35                     while( not dirChoice in range(len(dirList)) ):
36                         try:
37                             dirChoice = int( input( "Choisissez un fichier a traiter et
38                             entrez son numero : " ) )
39                         except NameError:
40                             print "Ceci n'est pas un nombre !"
41                     print "Dossier choisi : ", dirList[dirChoice]
42                     fileOk = True
43                     numeroTraitement = 0
44                     filesList = db.printFilesList(dirList[dirChoice])
45                     print filesList
```

```

44         action = int( input( "A partir de quelle action souhaitez-vous agir ?\n0-
        Tout\n1-Filtre passe-haut\n2-Fenetre de Hann\n3-Transformee de
        Fourier Rapide\n4-Fonction Mel\n5-Creation de la liste Mel\n6-
        Transformee de Fourier inverse\n7-Creation de vecteurs\n " ) )
45     for f in filesList:
46         dirName = os.path.dirname(f)
47         m = db.getWaveFile(f)
48         if action == 2:
49             content = db.getFile("handling/passe_haut_" + dirName + "_" + str
                (numeroTraitement) + ".txt")
50         elif action == 3:
51             content = db.getFile("handling/hann_" + dirName + "_" + str(
                numeroTraitement) + ".txt")
52         elif action == 4:
53             content = db.getFile("handling/fft_" + dirName + "_" + str(
                numeroTraitement) + ".txt")
54         elif action == 5:
55             content = db.getFile("handling/mel_" + dirName + "_" + str(
                numeroTraitement) + ".txt")
56         elif action == 6:
57             content = db.getFile("handling/mel_tab_" + dirName + "_" + str(
                numeroTraitement) + ".txt")
58         elif action == 7:
59             content = db.getFile("handling/fft_inverse_" + dirName + "_" +
                str(numeroTraitement) + ".txt")
60         else:
61             content = m[1]
62         mot,log = handlingOneWord(content,db,dirName,numeroTraitement)
63         if verbose:
64             print log
65         fileOk = False
66         numeroTraitement+=1
67         print "Mot reconnu pour " + f + ": ", mot
68
69
70     #####
71     ###          TEST GLOBAL          ###
72     #####
73     elif choice == 3:
74         fileName = recorder(db,"tmp",1,False,2,1)
75         cutBeginning( Db.prefixPath + "waves/tmp/", fileName + ".wav", "cut_" )
76         syncFile( Db.prefixPath + "waves/tmp/", "cut_" + fileName + ".wav", "sync_" )
77         db.addFileToList("tmp/sync_cut_" + fileName + ".wav", "waves/")
78         finalTest("tmp/sync_cut_" + fileName + ".wav")
79
80     #####
81     ###          RESULTATS INTERMEDIAIRES          ###
82     #####
83     elif choice == 4:
84         print "Voici la liste des mots a etudier : "
85         dirList = db.printDirFiles("storage/handling/")
86         dirChoice = -1
87         while( not dirChoice in range(len(dirList)) ):
88             try:
89                 dirChoice = int( input( "Choisissez un fichier a traiter et entrez
                son numero : " ) )
90             except NameError:
91                 print "Ceci n'est pas un nombre !"
92         print "Fichier choisi : ", dirList[dirChoice]
93         amp = db.getFile("handling/" + str(dirChoice))
94         db.addWaveFromAmp("output/" + str(dirChoice) + ".wav",44100,amp,"output/",
            False)
95
96
97     #####
98     ###          GESTION DES FICHIERS DE LA BDD          ###

```

```

99 #####
100 elif choice == 5:
101     choice3 = -1
102     while( not choice3 in range(1,6) ):
103         try:
104             choice3 = int(input("Que voulez-vous faire ?\n1-Supprimer un fichier\
n2-Supprimer un wav\n3-Synchroniser la BDD\n4-Synchroniser les
wav\n5-Synchroniser tous les fichiers\n"))
105         except NameError:
106             print "Ceci n'est pas un nombre !"
107     if choice3 == 1:
108         print "Fichiers : "
109         filesList = db.printDirFiles()
110         dirName = "storage/"
111     elif choice3 == 2:
112         print "Dossiers des waves : "
113         filesList = db.printDirFiles("waves/")
114         dirName = "waves/"
115     elif choice3 == 3:
116         db.sync()
117         db.sync("", "waves/")
118     elif choice3 == 4:
119         print "Voici la liste des mots a etudier : "
120         dirList = db.printDirFiles("waves/")
121         dirChoice = -1
122         while( not dirChoice in range(len(dirList)) ):
123             try:
124                 dirChoice = int( input( "Quel mot souhaitez vous traiter?: " ) )
125             except NameError:
126                 print "Ceci n'est pas un nombre !"
127             print "Dossier choisi : ", dirList[dirChoice]
128             filesList = db.printFilesList(dirList[dirChoice])
129             for f in filesList:
130                 cutBeginning( Db.prefixPath + "waves/", f, "" )
131                 syncFile( Db.prefixPath + "waves/", f, "" )
132     elif choice3 == 5:
133         db.sync()
134         db.sync("", "waves/")
135
136 #####
137 ###      CREATION D'UN HMM      ###
138 #####
139 elif choice == 6:
140     fileOk = False
141     while not fileOk:
142         #On choisit le dossier a afficher
143         print "Voici la liste des mots a etudier : "
144         dirList = db.printDirFiles("waves/")
145         dirChoice = -1
146         while( not dirChoice in range(len(dirList)) ):
147             try:
148                 dirChoice = int( input( "Choisissez un fichier a traiter et
entrez son numero : " ) )
149             except NameError:
150                 print "Ceci n'est pas un nombre !"
151             print "Dossier choisi : ", dirList[dirChoice]
152             fileOk = True
153             filesList = db.printFilesList(dirList[dirChoice])
154             if len(filesList) < 6:
155                 print "Pas assez d'enregistrements"
156                 continue
157             listVectors = []
158             numeroTraitement = 0
159             for f in filesList:
160                 dirName = os.path.dirname(f)
161                 m = db.getWaveFile(f)

```

```

162         content, log = handlingRecording(m[1], db, dirName, numeroTraitement)
163         listVectors.append(content)
164         fileOk = False
165         numeroTraitement += 1
166         print "Sauvegarde :"
167         db.addFile(dirList[dirChoice] + ".txt", listVectors, "hmm/")
168         hmmList = db.getFile("hmmList.txt")
169         if hmmList.get(dirList[dirChoice]):
170             hmmList[dirList[dirChoice]].append(dirList[dirChoice] + ".txt")
171         else:
172             hmmList[dirList[dirChoice]] = [dirList[dirChoice] + ".txt"]
173         print "Extraction :"
174         db.addFile("hmmList.txt", hmmList)
175         buildHMMs(hmmList.keys(), hmmList.values(), 500, Db.prefixPath + "hmm/")
176         saveHMMs(Db.prefixPath + "hmm/save.hmm")
177 #####
178 ###          LISTER LES HMMs          ###
179 #####
180 elif choice == 7:
181     hmmList = db.getFile("hmmList.txt")
182     print hmmList
183
184 def handlingOneWord(content, db, dirChoice, numeroTraitement, action=0):
185     """ Fait le traitement d'un mot pour en construire les vecteurs de Markov et
186         tester ensuite la compatibilite avec les automates existants
187         Retourne un tuple (motLePlusCompatible, log) """
188     content, log = handlingRecording(content, db, dirChoice, numeroTraitement, action)
189     loadHMMs(Db.prefixPath + "hmm/save.hmm")
190     return recognize(content), log
191
192 def handlingRecording(content, db, dirChoice, numeroTraitement, action=0):
193     log = ""
194     if action <= 1:
195         log += "Filtre passe-haut en cours...\n"
196         content = passe_haut(content)
197         log += "Filtre passe-haut termine...\n"
198         #db.addFile("handling/passe_haut_" + str(dirChoice) + "_" + str(
199             numeroTraitement) + ".txt", content)
200         #db.addWaveFromAmp("tmp/bob.wav", 44100, content)
201         log += "Sauvegarde effectuee...\n\n"
202     if action <= 2:
203         log += "Fenetre de Hann en cours...\n"
204         content = hann_window(content)
205         log += "Fenetre de Hann terminee...\n"
206         #db.addFile("handling/hann_" + str(dirChoice) + "_" + str(numeroTraitement) +
207             ".txt", content)
208         log += "Sauvegarde effectuee...\n\n"
209     if action <= 3:
210         log += "Transformee de Fourier rapide en cours...\n"
211         content = fftListe(content, True)
212         energyTable = construitTableauEnergy(content)
213         for k in range(len(content)):
214             for l in range(len(content[k])):
215                 content[k][l] = abs(content[k][l])
216         log += "Transformee de Fourier rapide terminee...\n"
217         #db.addFile("handling/fft_" + str(dirChoice) + "_" + str(numeroTraitement) +
218             ".txt", content)
219         log += "Sauvegarde effectuee...\n\n"
220     if action <= 4:
221         log += "Application de la fonction Mel en cours..."
222         for k in range(len(content)):
223             content[k] = fct_mel_pas(content[k], 10)
224         log += "Application de la fonction Mel terminee..."

```

```

223         db.addFile("handling/mel_" + str(dirChoice) + "_" + str(numeroTraitement) +
224                    ".txt", content)
225         log += "Sauvegarde effectuee...\n"
226     if action <= 5:
227         log += "Construction de la liste Mel en cours..."
228         for k in range(len(content)):
229             content[k] = mel_tab(content[k], 10)
230         log += "Construction de la liste Mel terminee..."
231         db.addFile("handling/mel_tab_" + str(dirChoice) + "_" + str(numeroTraitement)
232                    + ".txt", content)
233         log += "Sauvegarde effectuee...\n"
234     """
235     if action <= 5:
236         log += "Application de la fonction Mel en cours...\n"
237         for k in range(len(content)):
238             content[k] = triangularFilter(content[k], RATE)
239         log += "Application de la fonction Mel terminee...\n"
240         #db.addFile("handling/mel_" + str(dirChoice) + "_" + str(numeroTraitement) +
241                    ".txt", content)
242         log += "Sauvegarde effectuee...\n\n"
243     if action <= 6:
244         log += "Transformee de Fourier inverse en cours...\n"
245         for k in range(len(content)):
246             content[k] = inverseDCTII(content[k])
247         log += "Transformee de Fourier inverse terminee...\n"
248         #db.addFile("handling/fft_inverse_" + str(dirChoice) + "_" + str(
249                    numeroTraitement) + ".txt", content)
250         log += "Sauvegarde effectuee...\n"
251     if action <= 7:
252         log += "Creation de vecteurs HMM en cours...\n"
253         content = creeVecteur(content, energyTable)
254         log += "Creation de vecteurs HMM terminee...\n"
255         #db.addFile("handling/vecteurs_" + str(dirChoice) + "_" + str(
256                    numeroTraitement) + ".txt", content)
257         log += "Sauvegarde effectuee...\n\n"
258     #db.logDump(str(dirChoice) + "_" + str(numeroTraitement), log)
259     #db.logDump(str(dirChoice) + "_" + str(numeroTraitement))
260     return content, log
261
262 #def handlingOneWord(content, db, dirChoice, numeroTraitement, action=0, hmmList=[]):
263 def finalTest(fileName = ""):
264     db = Db("../db/", verbose=False)
265     fileOk = False
266     while not fileOk:
267         #On choisit le dossier a afficher
268         if fileName == "":
269             print "Voici la liste des mots a etudier : "
270             dirList = db.printDirFiles("waves/")
271             dirChoice = -1
272             while( not dirChoice in range(len(dirList)) ):
273                 try:
274                     dirChoice = int( input( "Choisissez un fichier a traiter et
275                                           entrez son numero : " ) )
276                 except NameError:
277                     print "Ceci n'est pas un nombre !"
278             print "Dossier choisi : ", dirList[dirChoice]
279             fileOk = True
280             numeroTraitement = 0
281             filesList = db.printFilesList(dirList[dirChoice])
282             print filesList
283             fileChoice = -1
284             while( not fileChoice in range(len(filesList)) ):
285                 try:
286                     fileChoice = int( input( "Choisissez un fichier a traiter et
287                                           entrez son numero : " ) )
288                 except NameError:

```



```

282         print "Ceci n'est pas un nombre !"
283     print "Fichier choisi : ", filesList[fileChoice]
284     n = fileChoice
285     f = filesList[fileChoice]
286     d = dirList[dirChoice]
287     fileOk = False
288 else:
289     f = fileName
290     fileOk = True
291     n = 1
292     d = ""
293     dirName = os.path.dirname(f)
294     m = db.getWaveFile(f)
295     mot,log = handlingOneWord(m[1],db,d,n)
296     print "Le mot reconnu est", mot
297     print "-----"
298 if __name__ == "__main__":
299     main(False)

```

A.2 server.py

```

1  if __name__ == '__main__':
2      import sys
3      if len(sys.argv) >= 2:
4          try:
5              PORT = int(sys.argv[1])
6          except TypeError:
7              print("Please provide an int !")
8      else:
9          PORT = 8010
10         print("Port set to default : %s" % PORT)
11
12     print("Launching server ...")
13     main.run(PORT)

```

A.3 gui.py

```

1  class Gui:
2      def __init__(self):
3          self.auth = AuthUser()
4          #self.auth.logIn("giliam", self.auth.hashPass("test"))
5          self.nbEnregistrement = 0
6          self.listeEnregistrements = []
7          self.db = Db("../db/")
8          self.fenetre3enabled = False
9          self.noiseOk = False
10
11     def ouverture(self):          #fonction qui ouvre une deuxieme fenetre graphique,
                                   et qui affiche le resultat
12         self.fenetre4=Tk()
13         self.fenetre4.attributes('-alpha', 1) #plein ecran
14         self.fenetre4.configure(background='white')
15         self.fenetre4.title("MIG SE 2013 - Liste des mots enregistres")
16         titre=Label(self.fenetre4, text='\nMIG SE 2013',font=("DIN", "34","bold"), fg
                                   ='#006eb8', bg="#ffffff")
17         titre.pack()
18         titre_logiciel=Label(self.fenetre4, text="Reconnaissance vocale\n\n\n\n",font
                                   =("DIN", "22"), bg="#ffffff")
19         titre_logiciel.pack()
20

```

```

21     panneau2=Label(self.fenetre4, text='Liste des mots actuellement reconnus :\n
    \n', font=("DIN", '14'), bg="#ffffff")
22     hmmList = self.db.getFile("hmmList.txt")
23     res = hmmList.keys()
24     resultat=Label(self.fenetre4, text="\n".join(res),font =("DIN", "28", "bold")
    , fg="#16d924", bg="#ffffff")
25     espace=Label(self.fenetre4, text="\n \n", bg="#ffffff")
26     panneau2.pack()
27     resultat.pack()
28     espace.pack()
29     bouton_fermer=Button(self.fenetre4,text='Quitter', command=self.fenetre4.
    destroy)
30     bouton_fermer.pack()
31     espace4=Label(self.fenetre4, text= ' \n ', bg="#ffffff")
32     espace4.pack()
33     self.fenetre4.mainloop()
34
35 def creationHmm(self):
36     self.bouton_enr.config(text="Terminer l'enregistrement du HMM", command=self.
    fenetre3.destroy)
37     self.noiseOk = False
38     listVectors = []
39     for l in self.listeEnregistrements:
40         content = self.db.getWaveFile(l)
41         content,log = handlingRecording(content[1],self.db,0,0,0)
42         listVectors.append(content)
43     hmmList = self.db.getFile("hmmList.txt")
44     if hmmList.get(self.mot):
45         hmmList[self.mot].append("client_" + self.mot + ".txt")
46     else:
47         hmmList[self.mot] = ["client_" + self.mot + ".txt"]
48     self.db.addFile( "hmmList.txt",hmmList )
49     self.db.addFile( "client_" + self.mot + ".txt", listVectors, "hmm/" )
50     buildHMMs(hmmList.keys(),hmmList.values(), 500, Db.prefixPath + "hmm/")
51     saveHMMs(Db.prefixPath + "hmm/save.hmm")
52
53 def enregistrer(self):
54     if self.nbEnregistrement == 0:
55         self.mot = self.saisirMot.get()
56     if not self.noiseOk:
57         self.errorMessage3.set("Vous n'avez pas encore enregistre le bruit !")
58     elif self.mot == "":
59         self.errorMessage3.set("Entrez un mot")
60     else:
61         self.errorMessage3.set("")
62         fileName = recorder(self.db,"tmp",1,False,1,confirm=False,fileName=self.
    mot + "_" + str( self.nbEnregistrement ) )
63         sox_handling(Db.prefixPath + "waves/tmp/" + self.mot + "_" + str(self.
    nbEnregistrement) + ".wav", Db.prefixPath + "waves/noise/" + self.mot
    + ".wav", Db.prefixPath + "waves/tmp/" )
64         cutBeginning(Db.prefixPath + "waves/tmp/", self.mot + "_" + str(self.
    nbEnregistrement) + ".wav", "cut_")
65         syncFile(Db.prefixPath + "waves/tmp/", self.mot + "_" + str(self.
    nbEnregistrement) + ".wav", "sync_")
66         self.listeEnregistrements.append("tmp/" + self.mot + "_" + str(self.
    nbEnregistrement) + ".wav")
67         self.nbEnregistrement += 1
68         self.bouton_enr.config(text="Lancer l'enregistrement numero " + str(self.
    nbEnregistrement + 1) )
69         if self.nbEnregistrement == NB_ITERATIONS:
70             self.creationHmm()
71
72
73 def enregistrerNoise(self):
74     self.mot = self.saisirMot.get()
75     if self.mot == "":

```

```

76         self.errorMessage3.set("Entrez un mot")
77     else:
78         self.errorMessage3.set("")
79         fileName = recorder(self.db,"noise",1,False,1,confirm=False,fileName=self
            .mot )
80         self.noiseOk = True
81
82     def loginAuth(self):
83         loginIn = self.loginC.get()
84         passwordIn = self.passwordC.get()
85         if passwordIn == "" or loginIn == "":
86             self.errorMessage.set("Il manque le pseudonyme ou le mot de passe !\n")
87         else:
88             self.auth.logIn(loginIn, self.auth.hashPass(passwordIn) )
89             self.fenetre2.destroy()
90             self.bouton_loginpopup.config(text='Se deconnecter')
91             self.errorMessage.set("")
92             Button(self.fenetre1,text='Liste des mots enregistres', command=self.
                ouverture).pack()
93             self.displayRecorder()
94
95     def registerAuth(self):
96         loginIn = self.loginR.get()
97         passwordIn = self.passwordR.get()
98         if passwordIn == "" or loginIn == "":
99             self.errorMessage.set("Il manque le pseudonyme ou le mot de passe !\n")
100        else:
101            if self.auth.getClient(loginIn) != "":
102                self.auth.newClient(loginIn, self.auth.hashPass(passwordIn), [])
103                self.errorMessage.set("Vous etes bien enregistre(e)\n")
104            else:
105                self.errorMessage.set("Le pseudonyme est deja utilise")
106
107     def fenetre3destroy(self):
108         self.fenetre3enabled = False
109         self.fenetre3.destroy()
110
111     def displayRecorder(self):
112         self.bouton_registerIn.pack()
113         if not self.fenetre3enabled:
114             self.fenetre3=Tk()
115             self.fenetre3.attributes('-alpha', 1) #plein ecran
116             self.fenetre3.configure(background='white')
117             self.fenetre3.title("MIG SE 2013 - Enregistrement")
118             titre=Label(self.fenetre3, text='\nMIG SE 2013',font =("DIN", "34","bold"
                ), fg='#006eb8',bg='#ffffff')
119             titre.pack()
120             titre_logiciel=Label(self.fenetre3, text="Reconnaissance vocale\n\n\n\n",
                font =("DIN", "22"),bg='#ffffff')
121             titre_logiciel.pack()
122             self.errorMessage3 = StringVar(self.fenetre3)
123
124             errorLegend=Label(self.fenetre3, textvariable=self.errorMessage3, fg='#
                B80002',bg='#ffffff')
125             errorLegend.pack()
126
127             demande_mot=Label(self.fenetre3, text="Entrez le mot que vous souhaitez
                enregistrer", font=("DIN", "14"),bg='#ffffff')
128             demande_mot.pack()
129             self.saisirMot=StringVar(self.fenetre3) # variable pour recevoir
                le texte saisi
130             saisieMot=Entry(self.fenetre3, textvariable=self.saisirMot, width=30,bg='
                #ffffff')
131             saisieMot.pack()
132             Label(self.fenetre3, text="Avant de proceder aux enregistrements
                necessaires, il convient d'enregistrer\

```

```

133 du bruit pour permettre efficacement le traitement du signalÂ \n", font=("DIN", '14'
    ),bg='#ffffff').pack()
134     self.bouton_bruit=Button(self.fenetre3, text='Enregistrer du bruit',
        command=self.enregistrerNoise, fg='#ff0000') #bouton qui enregistre
        et ouvre une nouvelle fenetre
135     self.bouton_bruit.pack()
136     Label(self.fenetre3, text="Il est necessaire pour creer un modele de
        Markov cache de proceder a une dizaine d'enregistrements du meme mot\
        n\
137 Vous aurez deux secondes a chaque enregistrement pour prononcer votre mot une fois 1
        'acquisition lancee\n", font=("DIN", '14'),bg='#ffffff').pack()
138     espace1=Label(self.fenetre3, text= ' \n ',bg='#ffffff')
139     espace1.pack()
140     self.bouton_enr=Button(self.fenetre3, text='Lancer 1\'enregistrement
        numero 1', command=self.enregistrer, fg='#ff0000') #bouton qui
        enregistre et ouvre une nouvelle fenetre
141     self.bouton_enr.pack()
142     bouton_fermer=Button(self.fenetre3,text='Quitter', command=self.
        fenetre3destroy)
143     bouton_fermer.pack()
144     espace4=Label(self.fenetre3, text= ' \n ',bg='#ffffff')
145     espace4.pack()
146     self.fenetre3enabled = True
147
148     def displayLogIn(self):
149
150         if self.auth.connected:
151             self.auth.logOut()
152             self.bouton_loginpopup.config(text='Se connecter / S\'inscrire')
153         else:
154             self.fenetre2 = Tk()
155             self.fenetre2.title("MIG SE 2013")
156             self.fenetre2.attributes('-alpha', 1)
157             self.fenetre2.configure(background='white')
158
159             self.loginC = StringVar(self.fenetre2)
160             self.passwordC = StringVar(self.fenetre2)
161             self.loginR = StringVar(self.fenetre2)
162             self.passwordR = StringVar(self.fenetre2)
163             self.errorMessage = StringVar(self.fenetre2)
164
165             errorLegend=Label(self.fenetre2, textvariable=self.errorMessage, fg='#
                B80002',bg='#ffffff')
166             errorLegend.pack()
167             Label(self.fenetre2, text="\nSe connecter \n", font=("DIN", '14'),bg='#
                ffffff').pack()
168
169             loginLabel=Label(self.fenetre2, text="Identifiant :", font=("DIN", '14'),
                bg='#ffffff')
170             loginLabel.pack()
171             loginForm=Entry(self.fenetre2, textvariable=self.loginC, width=30)
172             loginForm.pack()
173
174             passwordLabel=Label(self.fenetre2, text="\nMot de passe :", font=("DIN",
                '14'),bg='#ffffff')
175             passwordLabel.pack()
176             passwordForm=Entry(self.fenetre2, textvariable=self.passwordC, width=30)
177             passwordForm.pack()
178
179             bouton_envoyer=Button(self.fenetre2, text='Se connecter', command=self.
                loginAuth, fg='#000000')
180             bouton_envoyer.pack()
181
182             Label(self.fenetre2, text="\nS'inscrire \n", font=("DIN", '14'),bg='#
                ffffff').pack()
183

```

```

184         loginRegisterLabel=Label(self.fenetre2, text="Identifiant : ", font=("DIN
185             ", '14'),bg='#ffffff')
186         loginRegisterLabel.pack()
187         loginRegisterForm=Entry(self.fenetre2, textvariable=self.loginR, width
188             =30)
189         loginRegisterForm.pack()
190
191         passwordRegisterLabel=Label(self.fenetre2, text="\nMot de passe :", font
192             =("DIN", '14'),bg='#ffffff')
193         passwordRegisterLabel.pack()
194         passwordRegisterForm=Entry(self.fenetre2, textvariable=self.passwordR,
195             width=30)
196         passwordRegisterForm.pack()
197
198         bouton_envoyer=Button(self.fenetre2, text='S\'inscrire', command=self.
199             registerAuth, fg='#000000')
200         bouton_envoyer.pack()
201
202     def main(self):
203         self.fenetre1=Tk()
204         self.fenetre1.title("MIG SE 2013")
205         self.fenetre1.attributes('-zoomed', 1)
206         self.fenetre1.configure(background='white')
207         titre=Label(self.fenetre1, text="\nMIG SE 2013",font =("DIN", "34","bold"),
208             fg='#006eb8',bg='#ffffff')
209         titre.pack()
210         titre_logiciel=Label(self.fenetre1, text="Reconnaissance vocale\n",font =("
211             DIN", "22"),bg='#ffffff')
212         titre_logiciel.pack()
213
214         if not self.auth.connected:
215             self.bouton_loginpopup=Button(self.fenetre1,text="Se connecter / S'
216                 inscrire", command=self.displayLogIn)
217             self.bouton_loginpopup.pack()
218             self.bouton_registerIn=Button(self.fenetre1,text="Enregistrer", command=
219                 self.displayRecorder)
220
221         else:
222             self.bouton_loginpopup=Button(self.fenetre1,text="Se deconnecter",
223                 command=self.displayLogIn)
224             self.bouton_loginpopup.pack()
225             self.bouton_registerIn=Button(self.fenetre1,text="Enregistrer", command=
226                 self.displayRecorder)
227             self.displayRecorder()
228             Button(self.fenetre1,text='Liste des mots enregistres', command=self.
229                 ouverture).pack()
230
231         bouton_fermer1=Button(self.fenetre1,text='Quitter', command=self.fenetre1.
232             destroy)
233         bouton_fermer1.pack()
234
235         self.recorderlabel=Label(self.fenetre1,bg='#ffffff')
236         self.recorderlabel.pack()
237         espace3=Label(self.fenetre1, text= ' \n ',bg='#ffffff')
238         espace3.pack()
239         photo=PhotoImage(file="speechapp/img/logomigSE.gif") #insertion de l'image
240         labl = Label(self.fenetre1, image=photo,bg='#ffffff')
241         labl.pack()
242         self.fenetre1.mainloop()
243         self.fenetre3.mainloop()
244
245     gui = Gui()
246     gui.main()

```

B. handling

B.1 fenetrehann.py

```
1 def hann_window_bis (signal):
2     k = 0
3     l = len(signal)
4     j = 0
5     liste = []
6     while (k < l/(ecart_fenetre*RATE) and j<((2*l)-(ecart_fenetre*RATE))):
7         L = []
8         for i in range(int(temps_fenetre*RATE)+1):
9             L.append(signal[k*int(ecart_fenetre*RATE)+i]* \
10                0.5 * (1 - np.cos(2 * (np.pi) * (float(i) / (RATE * temps_fenetre)))))
11             j += 1
12         liste.append(L)
13         k += 1
14     return liste
15
16 def hann_window(signal):
17     l = len(signal)
18     k = 0
19     liste = []
20     while (k < l / (ecart_fenetre * RATE)):
21         L = []
22         for i in range(int(temps_fenetre * RATE) + 1):
23             try:
24                 L.append(signal[k * int(ecart_fenetre * RATE) + i] *
25                    0.5 \
26                    *(1 - np.cos(2 * np.pi * float(i) / (RATE * temps_fenetre))))
27             except IndexError:
28                 k = l/(ecart_fenetre*RATE)
29                 break
30             liste.append(L)
31             k+=1
32     return liste
33
34 if __name__ == "__main__":
35     #test de verification
36     z = hann_window([250.*i/100000 for i in range (88200)])
37     for i in range(len(z)):
38         print(len(z[i]))
```

B.2 inverseDCT.py

```
1 TAILLE_TABLEAU_MEL_ENTREE = 24
2
3 NOMBRE_COMPOSANTES_GARDEES = 13
4
5 B = TAILLE_TABLEAU_MEL_ENTREE
6
7
```

```

8  def inverseDCTI(x): # x represente le tableau en mel donne par les fonctions
    precedentes
9      X = np.zeros(B)
10     for k in range(B):
11         X[k] = (0.5*(x[0]+math.pow(-1, k)*x[B-1]) + reduce(add, [x[n]*math.
            cos(math.pi*n*k/(B-1)) for n in range(1,B-1)]))*math.sqrt(2./(B
            -1))
12     return X
13
14 def inverseDCTII(x):
15     X = [0 for i in range(B)]
16     X[0] = reduce(add, [x[n] for n in range(B)])/math.sqrt(B)
17     for k in range(1, B):
18         X[k]= reduce(add, [x[n]*math.cos(math.pi*(n+0.5)*k/B) for n in range(
            B)])*math.sqrt(2./B)
19     return X
20
21 def inverseDCTIII(x):
22     X = np.zeros(B)
23     for k in range(B):
24         X[k] = (0.5*x[0]+reduce(add, [x[n]*math.cos(math.pi*n*(k+0.5)/B) for
            n in range(1,B)]))*math.sqrt(2./B)
25     return X
26
27 if __name__ == "__main__":
28     a = [math.cos(i) for i in range(24)]
29     print(inverseDCTI(a))
30     print(inverseDCTII(a))
31     print(inverseDCTIII(a))

```

B.3 triangularFilterbank.py

```

1  def mel(f):
2      return 2595*math.log(1+f/700.)/math.log(10)
3
4  def triangularFilter(tab,FE):
5      """ Prend en parametre une fenetre de Hamming et la frequence d'
        echantillonnage et retourne la fenetre de Mel """
6      pasOutput = mel(FE/2.)/12.
7      pasFFT = FE/(2.*len(tab))
8      outputTab = [0 for i in range(24)]
9      for n in range(24):
10         debut = n/2 * pasOutput + (n%2)*pasOutput/2.
11         fin = n/2*pasOutput + (n%2)*pasOutput/2.+pasOutput
12         milieu = (debut+fin)/2.
13         for k in range(len(tab)):
14             f = k*pasFFT
15             if(mel(f) > debut and mel(f)<milieu):
16                 outputTab[n]+= abs(((mel(f)-debut)/(pasOutput/2.))*
                    tab[k])
17             elif(mel(f) > milieu and mel(f) < fin):
18                 outputTab[n]+= abs(((mel(f)- fin)/(pasOutput/2.) + 1)
                    *tab[k])
19             elif(mel(f) > fin):
20                 break
21     for n in range(24):
22         outputTab[n] = math.log(outputTab[n])
23     return outputTab

```

B.4 passehaut.py

```

1 def passe_haut (X):
2     N = len(X)
3     Y = zeros(N, float)
4     for i in range(1, N):
5         Y[i] = X[i] - 0.95 * X[i - 1]
6     Y[0] = X[0]
7     return(Y)

```

B.5 fft.cpp

```

1  typedef std::complex<double> cDouble;
2
3  cDouble* listToTab(boost::python::list l)
4  {
5      int N = boost::python::len(l);
6      cDouble *t = (cDouble*)malloc(N*sizeof(cDouble));
7      for (int i=0;i<N;i++)
8          t[i] = boost::python::extract<cDouble>(l[i]);
9      return t;
10 }
11
12 cDouble** listOfListToTab(boost::python::list l)
13 {
14     int N = boost::python::len(l);
15     cDouble** t = (cDouble**)malloc(N*sizeof(cDouble*));
16     for (int i=0;i<N;i++)
17         t[i] = listToTab(boost::python::extract<boost::python::list>(l[i]));
18     return t;
19 }
20
21 bool is2Power(int N) { return N==1 || (N%2==0 && is2Power(N/2)); }
22
23 int get2Power(int N) { return pow(2, ceil(log(N)/log(2))); }
24
25 cDouble e(int k, int N) { return exp((cDouble)(-2j*M_PI*k/N)); }
26
27 boost::python::list tabToList(cDouble *t, int N)
28 {
29     boost::python::list l = boost::python::list();
30     for (int i=0;i<N;i++)
31         l.append(t[i]);
32     return l;
33 }
34
35 cDouble* fftCT(cDouble *sig)
36 {
37     int i,j,k,p=0,f=1;
38     int N = 1024;
39     cDouble ekN;
40     cDouble **tmp = (cDouble**)malloc(2*sizeof(cDouble*));
41
42     for (i=0;i<2;i++)
43         tmp[i] = (cDouble*)malloc(N*sizeof(cDouble));
44     for (i=0;i<N;i++)
45         tmp[0][i] = sig[i];
46     for (i=N/2;i!=1;i/=2)
47     {
48         for (j=0;j<i;j++)
49             for (k=0;k<N/(2*i);k++)
50             {
51                 ekN = e(k,N/i)*tmp[p][i*(2*k+1)+j];
52                 tmp[f][i*k+j] = tmp[p][i*(2*k)+j] + ekN;

```



```

53         tmp[f][i*k+j+N/2] = tmp[p][i*(2*k)+j] - ekN;
54     }
55     p = f;
56     f = (p+1)%2;
57 }
58
59     free(tmp[f]);
60
61     return tmp[p];
62 }
63
64 cDouble* fft(cDouble *sig, int N, int *sizeC, bool mid)
65 {
66     cDouble* C;
67     if (is2Power(N)) {
68         C = fftCT(sig);
69     } else {
70         //std::cout<<"zPad needed";
71         int Npadded = get2Power(N);
72         cDouble* sigPadded = (cDouble*)malloc(Npadded*sizeof(cDouble));
73         for (int i=0;i<N;i++)
74             sigPadded[i] = sig[i];
75         for (int i=N;i<Npadded;i++)
76             sigPadded[i] = 0;
77         C = fftCT(sigPadded);
78         free(sigPadded);
79     }
80
81     if (mid) {
82         int n = 512;
83         cDouble *rep = (cDouble*)malloc(n*sizeof(cDouble));
84         for (int i=0;i<n;i++)
85             rep[i] = C[i];
86         *sizeC = n;
87         free(C);
88         return rep;
89     } else {
90         *sizeC = N;
91         return C;
92     }
93 }
94
95 boost::python::list fftListe(boost::python::list pyEchs, bool mid=true)
96 {
97     int nbEchs = boost::python::len(pyEchs);
98     cDouble **echs = listOfListToTab(pyEchs);
99     boost::python::list rep = boost::python::list();
100
101     int sizeC;
102     cDouble* C;
103
104     for (int i=0;i<nbEchs-1;i++)
105     {
106         if (i%5==0) {
107             //std::cout<<"Traitement du " << i << "eme echantillon..." <<
108                 std::endl;
109         }
110         C = fft(echs[i], 1024, &sizeC, mid);
111         rep.append(tabToList(C, sizeC));
112     }
113
114     //std::cout << "Traitement du dernier echantillon..." << std::endl;
115     int sizeLastEch = boost::python::len(pyEchs[nbEchs-1]);
116     C = fft(echs[nbEchs-1], sizeLastEch, &sizeC, mid);
117     rep.append(tabToList(C, sizeC));
118     free(C);

```

```
118 |
119 |     //std::cout << "Done !" << std::endl;
120 |     return rep;
121 | }
122 |
123 | BOOST_PYTHON_MODULE(fft)
124 | {
125 |     using namespace boost::python;
126 |     def("fftListe", fftListe);
127 | }
```

C. HMM

C.1 creationVecteurHMM.py

```
1 D = TAILLE_FINALE_MFCC
2
3 #tabMel = [[(i+1)*(k+1) for i in range(D)] for k in range(3)] #liste des tableaux de
   mel de l'echantillon sonore
4
5 #print tabMel
6
7 def creeVecteur(tabMel, energyTable):
8     choice = -1
9     output = [[0 for i in range(D)] for k in range(len(tabMel))]
10    for t in range(len(tabMel)):
11        output[t] = tabMel[t][0:D]
12        output[t][D-1] = energyTable[t]
13    while( not choice in range(2) ):
14        try:
15            choice = 1
16            #choice = int( input( "Voulez-vous incorporer les differences
   premieres ? \n 0 : non    1 : oui : " ))
17        except NameError:
18            print "Choix non valable"
19    if (choice == 1):
20        delta = [[0 for k in range(D)] for i in range(len(tabMel))]
21        choice = -1
22        for t in range(1,len(tabMel)):
23            for k in range(D):
24                delta[t][k] = output[t][k]-output[t-1][k]
25        for t in range(1, len(tabMel)):
26            for k in range(D):
27                output[t][k] += delta[t][k]
28        #print "output : "
29        #print output
30        #print "delta : "
31        #print delta
32        while( not choice in range(2) ):
33            try:
34                choice = 1
35                #choice = int( input( "Voulez-vous incorporer
   les differences secondes ? \n 0 : non
   1 : oui " ) )
36            except NameError:
37                print "Choix non valable"
38        if (choice == 1):
39            for t in range(1,len(tabMel)):
40                for k in range(D):
41                    output[t][k] += delta[t][k]-
   delta[t-1][k]
42            break
43        break
44    return output
```

C.2 markov.py

```
1 def getData(name):
2     with open(name, "r") as f:
3         content = pickle.Unpickler(f).load()
4
5     return content
6
7 def writeData(name, data):
8     with open(name, "w") as f:
9         pickle.Pickler(f).dump(data)
10
11 def getID(d):
12     l = []
13     for i in range(d):
14         l.append([])
15         for j in range(d):
16             if i==j:
17                 l[i].append(1.)
18             else:
19                 l[i].append(0.)
20
21     return l
22
23 def uniformPI(n):
24     PI = []
25     for i in range(n):
26         PI.append(1./n)
27     return PI
28
29 def uniformA(n):
30     A = []
31     for i in range(n):
32         A.append(uniformPI(n))
33     return A
34
35 def uniformC(n, m):
36     C = []
37     for i in range(n):
38         C.append(uniformPI(m))
39     return C
40
41 def uniformG_sigma(n, m, d):
42     G_sigma = []
43     for i in range(n):
44         G_sigma.append([])
45         for j in range(m):
46             G_sigma[i].append(getID(d))
47     return G_sigma
48
49 def normalize(l):
50     t = 0
51     for i in range(len(l)):
52         t += l[i]*l[i]
53     t = math.sqrt(t)
54
55     if t == 0:
56         return l
57
58     for i in range(len(l)):
59         l[i] /= t
60
61     return l
62
63 def distL(a, b):
```

```

64     r = 0
65     for i in range(len(a)):
66         r += (a[i]-b[i])*(a[i]-b[i])
67
68     return math.sqrt(r)
69
70 def coupures(l):
71     l_ = []
72     for i in range(len(l)-1):
73         l_.append(distL(l[i], l[i+1]))
74
75     normalize(l_)
76
77     coupures = []
78     for i in range(len(l)-1):
79         if l_[i] >= 0.35:
80             coupures.append(i)
81
82     return coupures
83
84 def add(l, l_):
85     for i in range(len(l_)):
86         done = False
87         for j in range(len(l)):
88             if l_[i] == l[j][0]:
89                 l[j] = (l[j][0], l[j][1]+1)
90                 done = True
91                 break
92             elif l_[i] < l[j][0]:
93                 l.insert(j, (l_[i], 1))
94                 done = True
95                 break
96         if done == False:
97             l.append((l_[i], 1))
98     return l
99
100 def convert(l):
101     max = 0
102     for i in range(len(l)):
103         if l[i][0] > max:
104             max = l[i][0]
105
106     l_ = [0 for k in range(max+1)]
107     for i in range(len(l)):
108         l_[l[i][0]] = l[i][1]
109
110     return l_
111
112 def spikes(l):
113     changed = True
114     while changed == True:
115         changed = False
116         l_ = [0 for k in range(len(l))]
117
118         if l[1] > l[0] and l[0] != 0:
119             l_[1] += l[0]
120             changed = True
121         else:
122             l_[0] += l[0]
123
124         for i in range(1, len(l)-1):
125             if l[i-1] > l[i] and l[i] != 0:
126                 l_[i-1] += l[i]
127                 changed = True
128             elif l[i+1] > l[i] and l[i] != 0:
129                 l_[i+1] += l[i]

```

```

130         changed = True
131     else:
132         l_[i] += l[i]
133
134     if l[len(l)-2] > l[len(l)-1] and l[len(l)-1] != 0:
135         l_[len(l)-2] += l[len(l)-1]
136         changed = True
137     else:
138         l_[len(l)-1] += l[len(l)-1]
139
140     l = l_
141
142     l_ = []
143
144     for i in range(len(l)):
145         if l[i] > 0:
146             l_.append(i)
147
148     return l_
149
150 def metaCoupures(seqs):
151     l = []
152     for i in range(len(seqs)):
153         add(l, coupures(seqs[i]))
154
155     l = convert(l)
156     l = spikes(l)
157
158     morceaux = [[] for k in range(len(l)+1)]
159     for i in range(len(seqs)):
160         k = 0
161         for j in range(len(seqs[i])):
162             morceaux[k].append(seqs[i][j])
163             if k < len(l) and j == l[k]:
164                 k += 1
165
166     mus = []
167     for i in range(len(morceaux)):
168         mu = [0 for k in range(len(morceaux[i][0]))]
169         for j in range(len(morceaux[i])):
170             for k in range(len(morceaux[i][j])):
171                 mu[k] += morceaux[i][j][k]
172
173         for k in range(len(morceaux[i][0])):
174             mu[k] /= len(morceaux[i])
175         mus.append([mu])
176
177     return mus
178
179 def buildHMMs(HMMs, HMMsPath, maxIt, path = "../db/hmm/"):
180     hmm.clearHMMs()
181
182     G_mu = []
183     seqs = []
184     for i in range(len(HMMsPath)):
185         seqs.append([])
186         G_mu.append([])
187         for j in range(len(HMMsPath[i])):
188             seqs[i].append(getData(path + HMMsPath[i][j]))
189             G_mu[i] = G_mu[i] + metaCoupures(seqs[i][j]) # FAIL : this does not work
190                 for multi speaker : G_mu should contain different mus for each
191                 speaker
192                 # FAIL : and the cutting of mus should be coherent
193
194     d = 13

```

```

194     for i in range(len(HMMs)):
195         n = len(G_mu[i])
196         m = len(HMMsPath[i])
197         hmm.createHMM(HMMs[i], HMMsPath[i], n, m, d, uniformPI(n), uniformA(n),
198                       uniformC(n, m), G_mu[i], uniformG_sigma(n, m, d))
199
200         passSeqs = []
201         for j in range(len(seqs[i])):
202             passSeqs = passSeqs + seqs[i][j]
203         x = hmm.baumWelch(HMMs[i], passSeqs, maxIt)
204         if x == 0.5:
205             print("HMM '{}' final likelihood (log) : -inf".format(HMMs[i]))
206             print("WARNING : HMM yielded 0 likelihood !")
207         elif x == 0.8:
208             print("ERROR : HMM '{}' not found !".format(HMMs[i]))
209         elif x >= 1:
210             print("HMM '{}' final likelihood (log) : {}".format(HMMs[i], 1-x))
211             print("WARNING : Baum-Welch algorithm ended because of iterations' limit
212                   ({}).format(maxIt))
213         else:
214             print("HMM '{}' final likelihood (log) : {}".format(HMMs[i], x))
215             print("")
216
217 def loadHMMs(fileName):
218     l = getData(fileName)
219     hmm.setHMMs(l)
220
221 def saveHMMs(fileName):
222     l = hmm.getHMMs()
223     writeData(fileName, l)
224
225 def recognize(seq):
226     l = hmm.recognize(seq)
227     print("Sequence recognized as : {} (log probability : {})".format(l[0], l[1]))
228     return l[0]
229
230 def recognizeList(name, path):
231     seqs = getData(path)
232     for i in range(len(seqs)):

```

C.3 tableauEnergyPerFrame.py

```

1 def construitTableauEnergy(content):
2     er = [0 for j in range(len(content))]
3     for k in range(len(content)):
4         er[k]=math.log(reduce(add,[abs(content[k][i])*abs(content[k][i]) for
5                                i in range(len(content))]))
6     return er

```

C.4 hmm.cpp

```

1 const long double MIN_VALUE = 0.00001;
2
3 long double det(long double **sigma, int d) {
4     long double r = 1;
5     for (int i = 0; i < d; i++)
6         r *= sigma[i][i];
7
8     return r;
9 }
10

```

```

11 long double calcProduct(long double **sigma, long double *mu, long double* x, int d)
12 {
13     long double r = 0;
14     for (int i = 0; i < d; i++)
15         r += (x[i]-mu[i])*(x[i]-mu[i])/sigma[i][i];
16
17     return r;
18 }
19
20 void sumVects(long double ***seqs, long double ****gammas, int d, int sN, int *sS,
21 int i, int k, long double *r) {
22     for (int a = 0; a < d; a++)
23         r[a] = 0;
24
25     for (int s = 0; s < sN; s++) {
26         for (int t = 0; t < sS[s]; t++) {
27             for (int a = 0; a < d; a++)
28                 r[a] += seqs[s][t][a]*gammas[s][t][i][k];
29         }
30     }
31 }
32
33 void sumMats(long double ***seqs, long double *mu, long double ****gammas, int d, int
34 sN, int *sS, int i, int k, long double **r) {
35     for (int a = 0; a < d; a++) {
36         for (int b = 0; b < d; b++)
37             r[a][b] = 0;
38
39         for (int s = 0; s < sN; s++) {
40             for (int t = 0; t < sS[s]; t++) {
41                 for (int a = 0; a < d; a++)
42                     r[a][a] += (seqs[s][t][a]-mu[a])*(seqs[s][t][a]-mu[a])*gammas[s][t][i][k];
43             }
44         }
45     }
46 }
47
48 void mulVect(long double *v, long double a, int d, long double *r) {
49     if (a == 0) { // Sum of sums is 0, so each sum is 0
50         for (int i = 0; i < d; i++)
51             r[i] = 200; // Far far far value so it's useless
52         return;
53     }
54
55     for (int i = 0; i < d; i++)
56         r[i] = v[i]/a;
57 }
58
59 void mulMat(long double **m, long double a, int d, long double **r) {
60     long double s = 0;
61     for (int i = 0; i < d; i++)
62         s += m[i][i]/a;
63
64     for (int i = 0; i < d; i++) {
65         for (int j = 0; j < d; j++) {
66             if (i == j) {
67                 if (m[i][j] <= MIN_VALUE) // Cap min values
68                     r[i][j] = MIN_VALUE;
69                 else
70                     r[i][j] = m[i][j]/a;
71             }
72             else
73                 r[i][j] = 0;
74         }
75     }
76 }

```



```

73 }
74
75 class ContinuousHMM {
76 public:
77     ContinuousHMM(std::string name, int n, int m, int d, std::vector<std::string>
        listSequences, long double *PI, long double **A, long double **C, long double
        ***G_mu, long double ****G_sigma);
78     ~ContinuousHMM();
79
80     void render();
81
82     long double calcGaussianValue(long double **sigma, long double *mu, long double *
        x); // Calculates a single probability for a vector x_ in the gaussian sigma
83     void calcProbabilitiesVector(long double *x, long double *r); // Calculate a
        probability for a vector x in each state's mixture
84     void calcProbabilitiesSequence(long double **seq, int s, long double **prob); //
        Calculate probabilities for each vector of the sequence seq
85     long double forward(long double **seq, int s, long double **prob, long double **
        alpha); // Implementation of the forward algorithm, returns the overall
        probability of the sequence
86     void backward(long double **seq, int s, long double **prob, long double **beta);
        // Implementation of the backward algorithm (doesn't return overall
        probability)
87     void calcXiOldGamma(long double **seq, int s, long double **alpha, long double **
        beta, long double p, long double **prob, long double ***xi, long double **
        oldGamma); // Calculates Xis and Old Gammas for latter calculus
88     void calcGamma(long double **seq, int s, long double **alpha, long double **beta,
        long double **prob, long double ***gamma); // Calculates gamma for latter
        calculus
89     void calcSums(long double ***seqs, int sN, int *sS, long double ****gammas, long
        double **littleSums, long double ***littleVect, long double ****littleMat,
        long double *fatSums); // Calculates partial sums for latter calculus
90     double baumWelch(long double ***seqs, int sN, int *sS, int maxIt = 100, int
        epsilon = 0.0000000001); // Baum-Welch Algorithm Implementation, learning
        algorithm
91
92     std::string name;
93     std::vector<std::string> listSequences;
94
95     int n;
96     int m;
97     int d;
98     long double *PI;
99     long double **A;
100    long double **C;
101    long double ***G_mu;
102    long double ****G_sigma;
103 };
104
105 ContinuousHMM::ContinuousHMM(std::string name, int n, int m, int d, std::vector<std::
    string> listSequences, long double *PI, long double **A, long double **C, long
    double ***G_mu, long double ****G_sigma) {
106     this->name = name;
107     this->n = n;
108     this->m = m;
109     this->d = d;
110     this->listSequences = listSequences;
111     this->PI = PI;
112     this->A = A;
113     this->C = C;
114     this->G_mu = G_mu;
115     this->G_sigma = G_sigma;
116 }
117
118 ContinuousHMM::~~ContinuousHMM() {
119 }

```

```

120
121 void ContinuousHMM::render() {
122     std::cout << "Markov's Continuous Automat : " << name << std::endl;
123     std::cout << "PI : [";
124     for (int i = 0; i < n; i++) {
125         if (i != n-1)
126             std::cout << PI[i] << ", ";
127         else
128             std::cout << PI[i] << "]" << std::endl << std::endl;
129     }
130
131     std::cout << "A : [" << std::endl;
132     for (int i = 0; i < n; i++) {
133         std::cout << "[";
134         for (int j = 0; j < n; j++) {
135             if (j != n-1)
136                 std::cout << A[i][j] << ", ";
137             else
138                 std::cout << A[i][j] << "]" << std::endl;
139         }
140         if (i == n-1)
141             std::cout << "]" << std::endl << std::endl;
142     }
143
144     std::cout << "C : [" << std::endl;
145     for (int i = 0; i < n; i++) {
146         std::cout << "[";
147         for (int j = 0; j < m; j++) {
148             if (j != m-1)
149                 std::cout << C[i][j] << ", ";
150             else
151                 std::cout << C[i][j] << "]" << std::endl;
152         }
153         if (i == n-1)
154             std::cout << "]" << std::endl << std::endl;
155     }
156
157     std::cout << "G_mu : [" << std::endl;
158     for (int i = 0; i < n; i++) {
159         std::cout << "[";
160         for (int j = 0; j < m; j++) {
161             if (j != m-1) {
162                 std::cout << "[";
163                 for (int a = 0; a < d; a++)
164                     std::cout << G_mu[i][j][a] << ", ";
165                 std::cout << "]" << std::endl;
166             } else {
167                 std::cout << "[";
168                 for (int a = 0; a < d; a++)
169                     std::cout << G_mu[i][j][a] << ", ";
170                 std::cout << "]" << std::endl;
171                 std::cout << "]" << std::endl;
172             }
173         }
174         if (i == n-1)
175             std::cout << "]" << std::endl << std::endl;
176     }
177
178     std::cout << "G_sigma : [" << std::endl;
179     for (int i = 0; i < n; i++) {
180         std::cout << "[";
181         for (int j = 0; j < m; j++) {
182             if (j != m-1) {
183                 std::cout << "[";
184                 for (int a = 0; a < d; a++)
185                     std::cout << G_sigma[i][j][a][a] << ", ";

```

```

186         std::cout << "]" << std::endl;
187     } else {
188         std::cout << "[";
189         for (int a = 0; a < d; a++)
190             std::cout << G_sigma[i][j][a][a] << ", ";
191         std::cout << "]" << std::endl;
192         std::cout << "]" << std::endl;
193     }
194 }
195 if (i == n-1)
196     std::cout << "]" << std::endl << std::endl;
197 }
198 }
199
200 long double ContinuousHMM::calcGaussianValue(long double **sigma, long double *mu,
201 long double *x) {
202     long double den = sqrt(pow(2*M_PI, d) * det(sigma, d));
203     long double num = exp((long double)-.5 * calcProduct(sigma, mu, x, d));
204
205     if (num/den > 1) // Probability over 1, Markov's bullshit continuous theory
206         return 1.1;
207     else
208         return num/den;
209 }
210
211 void ContinuousHMM::calcProbabilitiesVector(long double *x, long double *r) {
212     for (int i = 0; i < n; i++) {
213         r[i] = 0;
214         for (int j = 0; j < m; j++)
215             r[i] += C[i][j]*calcGaussianValue(G_sigma[i][j], G_mu[i][j], x);
216     }
217 }
218
219 void ContinuousHMM::calcProbabilitiesSequence(long double **seq, int s, long double
220 **prob) {
221     for (int i = 0; i < s; i++)
222         calcProbabilitiesVector(seq[i], prob[i]);
223 }
224
225 long double ContinuousHMM::forward(long double **seq, int s, long double **prob, long
226 double **alpha) {
227     for (int i = 0; i < n; i++) // Setting for each state value at t=0
228         alpha[0][i] = PI[i]*prob[0][i];
229
230     for (int t = 1; t < s; t++) {
231         for (int i = 0; i < n; i++) {
232             long double r = 0;
233             for (int j = 0; j < n; j++)
234                 r += alpha[t-1][j]*A[j][i];
235             alpha[t][i] = r*prob[t][i];
236         }
237     }
238
239     long double p = 0;
240     for (int i = 0; i < n; i++)
241         p += alpha[s-1][i];
242
243     return p;
244 }
245
246 void ContinuousHMM::backward(long double **seq, int s, long double **prob, long
247 double **beta) {
248     for (int i = 0; i < n; i++) // Setting for each state value at t=s-1
249         beta[s-1][i] = 1;
250
251     for (int t = s-2; t >= 0; t--) {

```

```

248     for (int i = 0; i < n; i++) {
249         long double r = 0;
250         for (int j = 0; j < n; j++)
251             r += A[i][j]*beta[t+1][j]*prob[t+1][j];
252         beta[t][i] = r;
253     }
254 }
255 }
256
257 // Uses xi and oldGamma. WARNING : xi and oldGamma must be defined !
258 void ContinuousHMM::calcXiOldGamma(long double **seq, int s, long double **alpha,
    long double **beta, long double p, long double **prob, long double ***xi, long
    double **oldGamma) {
259     for (int t = 0; t < s-1; t++) {
260         for (int i = 0; i < n; i++) {
261             oldGamma[t][i] = 0;
262             for (int j = 0; j < n; j++) {
263                 xi[t][i][j] = alpha[t][i]*A[i][j]*prob[t+1][j]*beta[t+1][j]/p;
264                 oldGamma[t][i] += xi[t][i][j];
265             }
266         }
267     }
268 }
269
270 void ContinuousHMM::calcGamma(long double **seq, int s, long double **alpha, long
    double **beta, long double **prob, long double ***gamma) {
271     for (int t = 0; t < s; t++) {
272         long double sumAB = 0;
273         for (int i = 0; i < n; i++)
274             sumAB += alpha[t][i]*beta[t][i];
275
276         for (int i = 0; i < n; i++) {
277             long double AB = alpha[t][i]*beta[t][i]/sumAB;
278             for (int k = 0; k < m; k++) {
279                 if (prob[t][i] == 0) // This means the sum of probabilities is 0,
                    thus a single one of them will be 0 too
280                     gamma[t][i][k] = 0;
281                 else
282                     gamma[t][i][k] = AB*C[i][k]*calcGaussianValue(G_sigma[i][k], G_mu
                        [i][k], seq[t])/prob[t][i];
283             }
284         }
285     }
286 }
287
288 // Uses littleSums, littleVect, littleMat, fatSums. WARNING : they must be defined !
289 void ContinuousHMM::calcSums(long double ***seqs, int sN, int *sS, long double ****
    gammas, long double **littleSums, long double ***littleVect, long double ****
    littleMat, long double *fatSums) {
290     for (int i = 0; i < n; i++) {
291         fatSums[i] = 0;
292         for (int k = 0; k < m; k++) {
293             littleSums[i][k] = 0;
294             for (int a = 0; a < d; a++) {
295                 littleVect[i][k][a] = 0;
296                 for (int b = 0; b < d; b++)
297                     littleMat[i][k][a][b] = 0;
298             }
299
300             for (int s = 0; s < sN; s++) {
301                 for (int t = 0; t < sS[s]; t++)
302                     littleSums[i][k] += gammas[s][t][i][k];
303             }
304             sumVects(seqs, gammas, d, sN, sS, i, k, littleVect[i][k]);
305             sumMats(seqs, G_mu[i][k], gammas, d, sN, sS, i, k, littleMat[i][k]);
306             fatSums[i] += littleSums[i][k];

```

```

307     }
308 }
309 }
310
311 double ContinuousHMM::baumWelch(long double ***seqs, int sN, int *sS, int maxIt, int
epsilon) {
312     long double oldLike = -1;
313     long double like = 1;
314     long double mean = 0;
315     long double rap = 1;
316     int it = 0;
317     //bool decrease = false;
318     int totalSize = 0;
319     for (int s = 0; s < sN; s++)
320         totalSize += sS[s];
321
322     // Allocation of new model parameters
323     long double *_PI = (long double*)malloc(sizeof(long double)*n);
324     long double **_A = (long double**)malloc(sizeof(long double)*n*n);
325     long double **_C = (long double**)malloc(sizeof(long double)*n*m);
326     long double ***_G_mu = (long double***)malloc(sizeof(long double)*n*m*d);
327     long double ****_G_sigma = (long double****)malloc(sizeof(long double)*n*m*d*d);
328     for (int i = 0; i < n; i++) {
329         _A[i] = (long double*)malloc(sizeof(long double)*n);
330         _C[i] = (long double*)malloc(sizeof(long double)*m);
331         _G_mu[i] = (long double**)malloc(sizeof(long double)*m*d);
332         _G_sigma[i] = (long double***)malloc(sizeof(long double)*m*d*d);
333         for (int j = 0; j < m; j++) {
334             _G_mu[i][j] = (long double*)malloc(sizeof(long double)*d);
335             _G_sigma[i][j] = (long double**)malloc(sizeof(long double)*d*d);
336             for (int a = 0; a < d; a++)
337                 _G_sigma[i][j][a] = (long double*)malloc(sizeof(long double)*d);
338         }
339     }
340
341     // Allocation of temporary arrays
342     long double ***alphas = (long double***)malloc(sizeof(long double)*totalSize*n);
343     long double ***betas = (long double***)malloc(sizeof(long double)*totalSize*n);
344     long double *ps = (long double*)malloc(sizeof(long double)*sN);
345     long double ***probs = (long double***)malloc(sizeof(long double)*totalSize*n);
346
347     long double ****xis = (long double****)malloc(sizeof(long double)*totalSize*n*n);
348     long double ***oldGammas = (long double***)malloc(sizeof(long double)*totalSize*n);
349     long double ****gammas = (long double****)malloc(sizeof(long double)*totalSize*n*
n);
350
351     for (int s = 0; s < sN; s++) {
352         alphas[s] = (long double**)malloc(sizeof(long double)*sS[s]*n);
353         betas[s] = (long double**)malloc(sizeof(long double)*sS[s]*n);
354         probs[s] = (long double**)malloc(sizeof(long double)*sS[s]*n);
355         xis[s] = (long double***)malloc(sizeof(long double)*sS[s]*n*n);
356         oldGammas[s] = (long double**)malloc(sizeof(long double)*sS[s]*n);
357         gammas[s] = (long double***)malloc(sizeof(long double)*sS[s]*n*n);
358         for (int t = 0; t < sS[s]; t++) {
359             alphas[s][t] = (long double*)malloc(sizeof(long double)*n);
360             betas[s][t] = (long double*)malloc(sizeof(long double)*n);
361             probs[s][t] = (long double*)malloc(sizeof(long double)*n);
362             xis[s][t] = (long double***)malloc(sizeof(long double)*n*n);
363             oldGammas[s][t] = (long double*)malloc(sizeof(long double)*n);
364             gammas[s][t] = (long double***)malloc(sizeof(long double)*n*n);
365             for (int i = 0; i < n; i++) {
366                 xis[s][t][i] = (long double*)malloc(sizeof(long double)*n);
367                 gammas[s][t][i] = (long double*)malloc(sizeof(long double)*n);
368             }
369         }

```

```

370 }
371
372 // Allocation of partial sums
373 long double **littleSums = (long double**)malloc(sizeof(long double)*n*m);
374 long double ***littleVect = (long double***)malloc(sizeof(long double)*n*m*d);
375 long double ****littleMat = (long double****)malloc(sizeof(long double)*n*m*d*d);
376 long double *fatSums = (long double*)malloc(sizeof(long double)*n);
377 for (int i = 0; i < n; i++) {
378     littleSums[i] = (long double*)malloc(sizeof(long double)*m);
379     littleVect[i] = (long double**)malloc(sizeof(long double)*m*d);
380     littleMat[i] = (long double***)malloc(sizeof(long double)*m*d*d);
381     for (int j = 0; j < m; j++) {
382         littleVect[i][j] = (long double*)malloc(sizeof(long double)*d);
383         littleMat[i][j] = (long double***)malloc(sizeof(long double)*d*d);
384         for (int a = 0; a < d; a++)
385             littleMat[i][j][a] = (long double*)malloc(sizeof(long double)*d);
386     }
387 }
388
389 long double r = 0;
390 long double num = 0;
391 long double den = 0;
392 while (it < maxIt) {
393     for (int s = 0; s < sN; s++)
394         calcProbabilitiesSequence(seqs[s], sS[s], probs[s]);
395
396     for (int s = 0; s < sN; s++) {
397         ps[s] = forward(seqs[s], sS[s], probs[s], alphas[s]);
398         backward(seqs[s], sS[s], probs[s], betas[s]);
399
400         calcXiOldGamma(seqs[s], sS[s], alphas[s], betas[s], ps[s], probs[s], xis[
401             s], oldGammas[s]);
402         calcGamma(seqs[s], sS[s], alphas[s], betas[s], probs[s], gammas[s]);
403     }
404
405     // Calculation of sums
406     calcSums(seqs, sN, sS, gammas, littleSums, littleVect, littleMat, fatSums);
407
408     for (int i = 0; i < n; i++) {
409         // Setting PI and A
410         r = 0;
411         den = 0;
412         for (int s = 0; s < sN; s++) {
413             r += oldGammas[s][0][i];
414             for (int t = 0; t < sS[s]-1; t++)
415                 den += oldGammas[s][t][i];
416         }
417         _PI[i] = r/sN;
418
419         if (den == 0) { // This means the probability to be in state i at time t
420             // So probabilities to go from i at t to j at t+1 will be 0 for each
421             // j
422             for (int j = 0; j < n; j++)
423                 _A[i][j] = (long double)(1./n); // Need to put equi-probabilities
424         }
425         else {
426             for (int j = 0; j < n; j++) {
427                 num = 0;
428                 for (int s = 0; s < sN; s++) {
429                     for (int t = 0; t < sS[s]-1; t++)
430                         num += xis[s][t][i][j];
431                 }
432                 _A[i][j] = num/den;
433             }
434         }
435     }
436 }

```

```

433
434     if (fatSums[i] == 0) { // C will be problematic
435         for (int k = 0; k < m; k++)
436             _C[i][k] = (long double)(1./m); // Need to put equi-probabilities
437     } else { // Setting C normally
438         for (int k = 0; k < m; k++)
439             _C[i][k] = littleSums[i][k]/fatSums[i];
440     }
441
442     // Setting G_mu and G_sigma
443     for (int k = 0; k < m; k++) {
444         mulVect(littleVect[i][k], littleSums[i][k], d, _G_mu[i][k]);
445         mulMat(littleMat[i][k], littleSums[i][k], d, _G_sigma[i][k]);
446     }
447 }
448
449 like = 1;
450 mean = 0;
451 for (int s = 0; s < sN; s++) {
452     like *= ps[s];
453     mean += ps[s];
454 }
455
456 mean /= sN;
457 //std::cout << "Likelihood : " << like << " (" << mean << ")" << std::endl <<
458     std::endl;
459 rap = 1;
460 if (oldLike != -1) {
461     rap = like/oldLike;
462     if (rap < 1) {
463         //decrease = true;
464         break;
465     }
466     else if (rap < (1+epsilon))
467         break;
468 }
469
470 oldLike = like;
471
472 for (int i = 0; i < n; i++) {
473     PI[i] = _PI[i];
474     for (int j = 0; j < n; j++)
475         A[i][j] = _A[i][j];
476     for (int k = 0; k < m; k++) {
477         C[i][k] = _C[i][k];
478
479         for (int a = 0; a < d; a++) {
480             G_mu[i][k][a] = _G_mu[i][k][a];
481             G_sigma[i][k][a][a] = _G_sigma[i][k][a][a];
482         }
483     }
484 }
485
486 it++;
487 }
488
489 double result = 0;
490 if (oldLike == 0)
491     result = 0.5;
492 else
493     result = log101(oldLike);
494
495 if (it == maxIt) {
496     //std::cout << "Ended on max iteration" << std::endl;
497     result = 1 - result;
498 }/* else {

```

```

498         if (decrease)
499             std::cout << "Ended on decreasing likelyhood" << std::endl;
500         else
501             std::cout << "Ended on stationary likelyhod" << std::endl;
502     }*/
503
504     //std::cout << "HMM '" << name << "' -> final likelyhood (iteration " << it << "
505     //    of " << maxIt << ") : " << oldLike << " and mean : " << mean << std::endl <<
506     //    std::endl;
507
508     // Freeings
509     // Freeing of new model parameters
510     for (int i = 0; i < n; i++) {
511         for (int j = 0; j < m; j++) {
512             for (int a = 0; a < d; a++)
513                 free(_G_sigma[i][j][a]);
514             free(_G_mu[i][j]);
515             free(_G_sigma[i][j]);
516         }
517         free(_A[i]);
518         free(_C[i]);
519         free(_G_mu[i]);
520         free(_G_sigma[i]);
521     }
522     free(_PI);
523     free(_A);
524     free(_C);
525     free(_G_mu);
526     free(_G_sigma);
527
528     // Freeing of temporary arrays
529     for (int s = 0; s < sN; s++) {
530         for (int t = 0; t < sS[s]; t++) {
531             for (int i = 0; i < n; i++) {
532                 free(xis[s][t][i]);
533                 free(gammas[s][t][i]);
534             }
535             free(alphas[s][t]);
536             free(betas[s][t]);
537             free(probs[s][t]);
538             free(xis[s][t]);
539             free(oldGammas[s][t]);
540             free(gammas[s][t]);
541         }
542         free(alphas[s]);
543         free(betas[s]);
544         free(probs[s]);
545         free(xis[s]);
546         free(oldGammas[s]);
547         free(gammas[s]);
548     }
549     free(alphas);
550     free(betas);
551     free(ps);
552     free(probs);
553     free(xis);
554     free(oldGammas);
555     free(gammas);
556
557     // Freeing of partial sums
558     for (int i = 0; i < n; i++) {
559         for (int j = 0; j < m; j++) {
560             for (int a = 0; a < d; a++)
561                 free(littleMat[i][j][a]);
562             free(littleVect[i][j]);
563             free(littleMat[i][j]);
564         }
565     }

```



```

562     }
563     free(littleSums[i]);
564     free(littleVect[i]);
565     free(littleMat[i]);
566 }
567 free(littleSums);
568 free(littleVect);
569 free(littleMat);
570 free(fatSums);
571
572 return result;
573 }
574
575 std::vector<ContinuousHMM*> HMMs;
576
577 int findHMM(std::string s) {
578     for (unsigned int i = 0; i < HMMs.size(); i++) {
579         if (HMMs.at(i)->name.compare(s) == 0)
580             return (int)i;
581     }
582
583     std::cout << "HMM " << s << " ' not found !" << std::endl;
584     return -1;
585 }
586
587 boost::python::list tabToList(long double *tab, int size) {
588     boost::python::list _list;
589     for (int i = 0; i < size; i++)
590         _list.append(tab[i]);
591
592     return _list;
593 }
594
595 boost::python::list tTabToLList(long double **tab, int size, int subSize) {
596     boost::python::list _list;
597     for (int i = 0; i < size; i++)
598         _list.append(tabToList(tab[i], subSize));
599
600     return _list;
601 }
602
603 boost::python::list tTTabToLList(long double ***tab, int size, int subSize, int
subSubSize) {
604     boost::python::list _list;
605     for (int i = 0; i < size; i++)
606         _list.append(tTabToLList(tab[i], subSize, subSubSize));
607
608     return _list;
609 }
610
611 boost::python::list tTTabToLList(long double ****tab, int size, int subSize, int
subSubSize, int subSubSubSize) {
612     boost::python::list _list;
613     for (int i = 0; i < size; i++)
614         _list.append(tTTabToLList(tab[i], subSize, subSubSize, subSubSubSize));
615
616     return _list;
617 }
618
619 void createHMM(boost::python::str _name, boost::python::list _listSequences, int n,
int m, int d, boost::python::list _PI, boost::python::list _A, boost::python::
list _C, boost::python::list _G_mu, boost::python::list _G_sigma) {
620     std::string name = boost::python::extract<std::string>(_name);
621     long double *PI = (long double*)malloc(sizeof(long double)*n);
622     long double **A = (long double**)malloc(sizeof(long double)*n*n);
623     long double **C = (long double**)malloc(sizeof(long double)*n*m);

```

```

624     long double ***G_mu = (long double***)malloc(sizeof(long double)*n*m*d);
625     long double ****G_sigma = (long double****)malloc(sizeof(long double)*n*m*d*d);
626
627     std::vector<std::string> listSequences;
628     int numSeqs = boost::python::len(_listSequences);
629     for (int i = 0; i < numSeqs; i++)
630         listSequences.push_back(boost::python::extract<std::string>(_listSequences[i]
631                               ));
632
633     for(int i = 0; i < n; i++) {
634         PI[i] = boost::python::extract<long double>(_PI[i]);
635
636         A[i] = (long double*)malloc(sizeof(long double)*n);
637         for (int j = 0; j < n; j++)
638             A[i][j] = boost::python::extract<long double>(_A[i][j]);
639
640         C[i] = (long double*)malloc(sizeof(long double)*m);
641         G_mu[i] = (long double**)malloc(sizeof(long double)*m*d);
642         G_sigma[i] = (long double***).malloc(sizeof(long double)*m*d*d);
643         for (int j = 0; j < m; j++) {
644             C[i][j] = boost::python::extract<long double>(_C[i][j]);
645
646             G_mu[i][j] = (long double*)malloc(sizeof(long double)*d);
647             G_sigma[i][j] = (long double**).malloc(sizeof(long double)*d*d);
648             for (int k = 0; k < d; k++) {
649                 G_mu[i][j][k] = boost::python::extract<long double>(_G_mu[i][j][k]);
650
651                 G_sigma[i][j][k] = (long double*)malloc(sizeof(long double)*d);
652                 for (int l = 0; l < d; l++)
653                     G_sigma[i][j][k][l] = boost::python::extract<long double>(_G_sigma[i][j][k][l]);
654             }
655         }
656     }
657
658     ContinuousHMM *M = new ContinuousHMM(name, n, m, d, listSequences, PI, A, C, G_mu,
659     , G_sigma);
660     HMMs.push_back(M);
661 }
662
663 void createHMMFromList(boost::python::list _HMM) {
664     boost::python::str _name = boost::python::extract<boost::python::str>(_HMM[0]);
665     boost::python::list _listSequences = boost::python::extract<boost::python::list>(_HMM[1]);
666     int n = boost::python::extract<int>(_HMM[2]);
667     int m = boost::python::extract<int>(_HMM[3]);
668     int d = boost::python::extract<int>(_HMM[4]);
669     boost::python::list _PI = boost::python::extract<boost::python::list>(_HMM[5]);
670     boost::python::list _A = boost::python::extract<boost::python::list>(_HMM[6]);
671     boost::python::list _C = boost::python::extract<boost::python::list>(_HMM[7]);
672     boost::python::list _G_mu = boost::python::extract<boost::python::list>(_HMM[8]);
673     boost::python::list _G_sigma = boost::python::extract<boost::python::list>(_HMM[9]);
674
675     createHMM(_name, _listSequences, n, m, d, _PI, _A, _C, _G_mu, _G_sigma);
676 }
677
678 void setHMMs(boost::python::list _HMMs) {
679     int size = boost::python::len(_HMMs);
680     for (int i = 0; i < size; i++) {
681         createHMMFromList(boost::python::extract<boost::python::list>(_HMMs[i]));
682     }
683 }
684
685 boost::python::list getHMMs() {

```

```

685     boost::python::list _HMMs;
686     for (unsigned int i = 0; i < HMMs.size(); i++) {
687         boost::python::list _HMM;
688         _HMM.append(boost::python::str(HMMs.at(i)->name));
689         boost::python::list _listSequences;
690         for (unsigned int j = 0; j < HMMs.at(i)->listSequences.size(); j++)
691             _listSequences.append(HMMs.at(i)->listSequences.at(j));
692         _HMM.append(_listSequences);
693         int n = HMMs.at(i)->n;
694         int m = HMMs.at(i)->m;
695         int d = HMMs.at(i)->d;
696         _HMM.append(n);
697         _HMM.append(m);
698         _HMM.append(d);
699         _HMM.append(tabToList(HMMs.at(i)->PI, n));
700         _HMM.append(tTabToLList(HMMs.at(i)->A, n, n));
701         _HMM.append(tTabToLList(HMMs.at(i)->C, n, m));
702         _HMM.append(tTTabToLLList(HMMs.at(i)->G_mu, n, m, d));
703         _HMM.append(tTTTabToLLList(HMMs.at(i)->G_sigma, n, m, d, d));
704
705         _HMMs.append(_HMM);
706     }
707
708     return _HMMs;
709 }
710
711 void clearHMMs() {
712     while (HMMs.size() != 0) {
713         ContinuousHMM *M = HMMs.at(0);
714         HMMs.erase(HMMs.begin());
715         delete M;
716     }
717 }
718
719 void removeHMM(boost::python::str _name) {
720     std::string name = boost::python::extract<std::string>(_name);
721     int id = findHMM(name);
722     if (id == -1)
723         return;
724
725     ContinuousHMM *M = HMMs.at(id);
726     HMMs.erase(HMMs.begin()+id);
727     delete M;
728 }
729
730 void renderHMM(boost::python::str _name) {
731     std::string name = boost::python::extract<std::string>(_name);
732     int id = findHMM(name);
733     if (id != -1)
734         HMMs.at(id)->render();
735 }
736
737 long double forward(boost::python::str _name, boost::python::list _seq) {
738     std::string name = boost::python::extract<std::string>(_name);
739     int id = findHMM(name);
740     if (id == -1)
741         return -1;
742
743     int s = boost::python::len(_seq);
744
745     long double **seq = (long double**)malloc(sizeof(long double)*s*HMMs.at(id)->d);
746     for (int t = 0; t < s; t++) {
747         seq[t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->d);
748         for (int i = 0; i < HMMs.at(id)->d; i++)
749             seq[t][i] = boost::python::extract<long double>(_seq[t][i]);
750     }

```

```

751
752     long double p;
753     long double **prob = (long double**)malloc(sizeof(long double)*s*HMMs.at(id)->n);
754     long double **alpha = (long double**)malloc(sizeof(long double)*s*HMMs.at(id)->n);
755
756     for (int t = 0; t < s; t++) {
757         prob[t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->n);
758         alpha[t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->n);
759     }
760
761     HMMs.at(id)->calcProbabilitiesSequence(seq, s, prob);
762     p = HMMs.at(id)->forward(seq, s, prob, alpha);
763
764     for (int t = 0; t < s; t++) {
765         free(seq[t]);
766         free(prob[t]);
767         free(alpha[t]);
768     }
769     free(seq);
770     free(prob);
771     free(alpha);
772
773     std::cout << "Forward : " << p << std::endl;
774     return p;
775 }
776
777 double baumWelch(boost::python::str _name, boost::python::list _seqs, int it) {
778     std::string name = boost::python::extract<std::string>(_name);
779     int id = findHMM(name);
780     if (id == -1)
781         return 0.8;
782
783     int sN = boost::python::len(_seqs);
784
785     int *sS = (int*)malloc(sizeof(int)*sN);
786     int totalSize = 0;
787     for (int s = 0; s < sN; s++) {
788         sS[s] = boost::python::len(boost::python::extract<boost::python::list>(_seqs[s]));
789         totalSize += sS[s];
790     }
791
792     long double ***seqs = (long double***)malloc(sizeof(long double)*totalSize*HMMs.at(id)->d);
793     for (int s = 0; s < sN; s++) {
794         seqs[s] = (long double**)malloc(sizeof(long double)*sS[s]*HMMs.at(id)->d);
795         for (int t = 0; t < sS[s]; t++) {
796             seqs[s][t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->d);
797             for (int i = 0; i < HMMs.at(id)->d; i++)
798                 seqs[s][t][i] = boost::python::extract<long double>(_seqs[s][t][i]);
799         }
800     }
801
802     double d = HMMs.at(id)->baumWelch(seqs, sN, sS, it);
803
804     for (int s = 0; s < sN; s++) {
805         for (int t = 0; t < sS[s]; t++) {
806             free(seqs[s][t]);
807         }
808         free(seqs[s]);
809     }
810     free(sS);
811
812     return d;
813 }

```

```

814
815 boost::python::list recognize(boost::python::list _seq) {
816     int s = boost::python::len(_seq);
817
818     int d = 13;
819     long double **seq = (long double**)malloc(sizeof(long double)*s*d);
820     for (int t = 0; t < s; t++) {
821         seq[t] = (long double*)malloc(sizeof(long double)*d);
822         for (int i = 0; i < d; i++)
823             seq[t][i] = boost::python::extract<long double>(_seq[t][i]);
824     }
825
826     long double p;
827     long double maxP = 0;
828     std::string maxName;
829     long double **prob;
830     long double **alpha;
831
832     for (int id = 0; id < (int)HMMs.size(); id++) {
833         prob = (long double**)malloc(sizeof(long double)*s*HMMs.at(id)->n);
834         alpha = (long double**)malloc(sizeof(long double)*s*HMMs.at(id)->n);
835         for (int t = 0; t < s; t++) {
836             prob[t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->n);
837             alpha[t] = (long double*)malloc(sizeof(long double)*HMMs.at(id)->n);
838         }
839
840         HMMs.at(id)->calcProbabilitiesSequence(seq, s, prob);
841         p = HMMs.at(id)->forward(seq, s, prob, alpha);
842
843         if (p > maxP) {
844             maxP = p;
845             maxName = HMMs.at(id)->name;
846         }
847
848         for (int t = 0; t < s; t++) {
849             free(prob[t]);
850             free(alpha[t]);
851         }
852         free(prob);
853         free(alpha);
854     }
855
856     for (int t = 0; t < s; t++)
857         free(seq[t]);
858     free(seq);
859
860     boost::python::list result;
861     result.append(maxName);
862     result.append(log10l(maxP));
863
864     return result;
865 }
866
867 BOOST_PYTHON_MODULE(hmm)
868 {
869     using namespace boost::python;
870     def("createHMM", createHMM);
871     def("setHMMs", setHMMs);
872     def("getHMMs", getHMMs);
873     def("clearHMMs", clearHMMs);
874     def("removeHMM", removeHMM);
875     def("renderHMM", renderHMM);
876     def("forward", forward);
877     def("baumWelch", baumWelch);
878     def("recognize", recognize);
879 }

```

D. recorder

D.1 recorder.py

```
1 def recorder(db, dirName="", nbRecording=-1, askForWord=True, seconds=-1, nbWords=1,
2   fileName="", confirm=True):
3     """ Procède a l'enregistrement """
4     if nbRecording < 0:
5         nbRecording = raw_input("Combien d'enregistrement par mots ? ")
6         nbRecording = int(nbRecording)
7     n = ""
8     for k in range(nbWords):
9         if askForWord:
10             mot = raw_input("Entrez le mot a enregistrer : ")
11         else:
12             mot = ""
13
14         if seconds < 0:
15             seconds = raw_input("Entrez le nombre de secondes pour l'
16                               enregistrement : ")
17         seconds = float(seconds) + 1
18
19         for i in range(nbRecording):
20             if confirm:
21                 raw_input("Appuyez sur une touche pour commencer l'
22                           enregistrement : ")
23             p = pyaudio.PyAudio()
24
25             stream = p.open(format=FORMAT,
26                             channels=CHANNELS,
27                             rate=RATE,
28                             input=True,
29                             frames_per_buffer=CHUNK)
30
31             #print "Enregistrement[" , i , "]:"
32
33             frames = []
34
35             for j in range(0, int(RATE / CHUNK * seconds)):
36                 data = stream.read(CHUNK)
37                 frames.append(data)
38
39             #print "Fin - Enregistrement[" , i , "]:"
40
41             stream.stop_stream()
42             stream.close()
43             p.terminate()
44             if dirName != "":
45                 random.seed()
46                 if fileName == "":
47                     fileName = hashlib.sha224(str(random.randint(
48                                     0,1e10))).hexdigest()
49                     name = dirName + "/" + fileName + ".wav"
50             else:
```

```

48         n = str(i)
49         name = mot + "/" + n + ".wav"
50         db.addWave(name, CHANNELS, p.get_sample_size(FORMAT), RATE,
                    frames, p)
51         #print "Fin du mot ", i
52     return fileName

```

D.2 sync.py

```

1  def sync(amplitudes):
2      tOut = 400
3      N = len(amplitudes)
4      coeff_lissage = 5
5      max = 0
6      for i in range(N):
7          if abs(amplitudes[i]) > max:
8              max = abs(amplitudes[i])
9
10         #print("Max is {}".format(max))
11
12         seuilFor = max/8
13         seuilBack = max/7
14         #print("Seuil is {}".format(valeurSeuil))
15
16         maxDiff = 300
17         maxRemove = 800
18         #print("MaxDiff is {}".format(maxDiff))
19
20         iMin = -1
21         iMin2 = -1
22         iMax = -1
23         iMax2 = -1
24         inIt = False
25         lastHit = -1
26
27         for i in range(N):
28             if iMin == -1 and amplitudes[i] > seuilFor:
29                 iMin = i
30                 lastHit = i
31                 inIt = True
32             if iMin != -1:
33                 if i - iMin > maxRemove: # Won't remove more than maxRemove
34                     break
35                 elif inIt == True and amplitudes[i] > seuilFor:
36                     lastHit = i
37                 elif inIt == True and amplitudes[i] < seuilFor and i - lastHit >= maxDiff:
38                     inIt = False
39                 elif inIt == False and amplitudes[i] > seuilFor:
40                     iMin2 = i
41                     break
42
43         inIt = False
44         lastHit = -1
45
46         for i in range(N-1, -1, -1):
47             if iMax == -1 and amplitudes[i] > seuilBack:
48                 iMax = i
49                 lastHit = i
50                 inIt = True
51             if iMax != -1:
52                 if iMax - i > maxRemove: # Won't remove more than maxRemove
53                     break

```

```

54         elif inIt == True and amplitudes[i] > seuilBack:
55             lastHit = i
56         elif inIt == True and amplitudes[i] < seuilBack and lastHit - i >=
             maxDiff:
57             inIt = False
58         elif inIt == False and amplitudes[i] > seuilBack:
59             iMax2 = i
60             break
61
62     #print("iMin is {}".format(iMin))
63     #print("iMin2 is {}".format(iMin2))
64     #print("iMax is {}".format(iMax))
65     #print("iMax2 is {}".format(iMax2))
66     #print("")
67
68     if iMin2 != -1 and iMin2-iMin > tOut:
69         iMin2 -= tOut
70     if iMax2 != -1 and iMax-iMax2 > tOut:
71         iMax2 += tOut
72
73     if iMin > tOut:
74         iMin -= tOut
75     if N-1 - iMax > tOut:
76         iMax += tOut
77
78
79     amplitudes_coupe = [0. for i in range(iMax-iMin+1)]
80     for i in range(iMax-iMin+1):
81         amplitudes_coupe[i] = amplitudes[iMin + i]
82
83     if iMin2 != -1 or iMax2 != - 1:
84         if iMin2 == -1:
85             iMin2 = iMin
86         if iMax2 == -1:
87             iMax2 = iMax
88
89         if iMin2 - iMin <= maxRemove and iMax - iMax2 <= maxRemove:
90             return amplitudes_coupe
91         amplitudes_coupe2 = [0. for i in range(iMax2-iMin2+1)]
92         for i in range(iMax2-iMin2+1):
93             amplitudes_coupe2[i] = amplitudes[iMin2 + i]
94
95         return sync(amplitudes_coupe2)
96     else:
97         return amplitudes_coupe
98 def syncFile(path, name, prefix = "sync_"):
99     #print("Synching : {}".format(name))
100     ampli = scipy.io.wavfile.read(path + name)
101     ampli2 = sync(ampli[1])
102     scipy.io.wavfile.write(path + prefix + name, ampli[0], int16(ampli2))
103     #print("Done\n\n")
104
105 def cutBeginning(path,name,prefix = "cut_"):
106     ampli = scipy.io.wavfile.read(path + name)
107     ampli2 = ampli[1][22050:]
108     scipy.io.wavfile.write(path + prefix + name, ampli[0], int16(ampli2))
109
110 def sox_handling(fileName, noiseName, pathToTmp = "../db/waves/tmp/"):
111     pass
112     #os.system('sox "' + noiseName + '" -n noiseprof "' + pathToTmp + 'noise.prof"')
113     #os.system('sox "' + fileName + '" "' + fileName + '" noisered "' + pathToTmp + '
         noise.prof" 0.21')
114     #os.remove(pathToTmp + "noise.prof")
115
116 if __name__ == "__main__":
117     syncFile("3_0")

```



```
118     syncFile("3_1")
119     syncFile("5_0")
120     syncFile("5_1")
```

E. utils

E.1 animate.py

```
1 data = Cs          # Liste de listes des valeurs
2 xMin = 350         # Echelles d'affichage
3 xMax = 550
4 yMin = 0
5 yMax = 100
6 interv = 50        # Millisecondes entre chaque image
7
8
9 fram = len(data)
10 fig = plt.figure()
11 ax = plt.axes(xlim=(xMin, xMax), ylim=(yMin, yMax))
12 line, = ax.plot([], [], lw=2)
13
14 def init():
15     line.set_data([], [])
16     return line,
17
18 def animate(i):
19     L = len(data[i])
20     x = np.linspace(0, L-1, L)
21     y = [abs(data[i][int(k)]) for k in x]
22     line.set_data(x, y)
23     return line,
24
25 anim = animation.FuncAnimation(fig, animate, init_func=init,
26                               frames=fram, interval=interv, blit=True)
```

E.2 constantes.py

```
1 # pour le recorder:
2 CHUNK = 1024 # nombre de bits enregistres par boucle
3 FORMAT = pyaudio.paInt16
4 CHANNELS = 1 # On est en mono
5 RATE = 44100 #Frequence
6
7 # Synchro:
8 COEFF_LISSAGE = 5 # a determiner empiriquement
9 T_MIN = 50 # blanc minimum avant le son
10 COEFF_COUPE = 0.0000001 # en pourcent
11
12 # fenetre hann :
13
14 ecart_fenetre = 0.01301587
15 temps_fenetre = 0.023219954648526
16
17 #Creation MFCC
18
19 TAILLE_FINALE_MFCC = 13
```

```
20
21 NB_ITERATIONS = 10
```

E.3 db.py

```
1 class Db:
2     """ Files manager to store .wav and hmm
3     Attributes :
4         -> filesList : name of the file containing the list of files stored """
5
6     filesListName = "filesList"
7     prefixPath = ""
8     verbose = False
9
10    def __init__(self, prefixPath = "", filesListName = "filesList", verbose = False)
11        :
12        """ Constructor which needs prefix of the directory which contains (or will
13            contain) the stored files """
14        Db.prefixPath = prefixPath
15        Db.filesListName = filesListName
16        Db.verbose = verbose
17        self.log = ""
18        try:
19            with open(Db.prefixPath + Db.filesListName + ".txt","r") as f:
20                self.filesList = pickle.Unpickler(f).load()
21        except IOError:
22            Db.reset(True)
23            raise Exception("L'instanciation a ete annulee car le fichier de gestion
24                de la base de donnees n'existe pas")
25
26
27
28
29
30
31    def getFile(self,fileName,dirFile=""):
32        """ Add a file to the list of files handled par the database system
33        Parameters :
34            @fileName : name of the file in the storage directory prefixed by
35                dirFile
36            @dirFile : add a prefix to files and give others handling available
37        """
38        if len(dirFile) == 0:
39            dirFile = "storage"
40        if fileName in self.filesList:
41            try:
42                if dirFile == "waves":
43                    content = scipy.io.wavfile.read(Db.prefixPath + dirFile + "/" +
44                        fileName)
45                    return content
46                elif fileName in self.filesList:
47                    with open(Db.prefixPath + dirFile + "/" + fileName,"r") as f:
48                        content = pickle.Unpickler(f).load()
49                    return content
50            except IOError:
51                raise Exception("La lecture de fichier a echoue")
52        else:
53            self.addLog("le fichier n'est pas gere pas la base de donnees")
54            return ""
55
56
57
58
59
60
61    def getWaveFile(self,fileName):
62        """ Alias of getFile for .wav """
63        return self.getFile(fileName,"waves")
```

```

54
55
56
57 def addWave(self, fileName, CHANNELS, FORMAT, RATE, frames, p):
58     if os.access(Db.prefixPath + "waves/" + fileName, os.F_OK):
59         pass
60         #Il faudrait rajouter la gestion de l'existence de deux mÃªmes fichiers
61     dirName = os.path.dirname(fileName)
62     if not os.access(Db.prefixPath + "waves/" + dirName, os.F_OK):
63         os.mkdir(Db.prefixPath + "waves/" + dirName)
64     wf = wave.open(Db.prefixPath + "waves/" + fileName, 'wb')
65     wf.setnchannels(CHANNELS)
66     wf.setsampwidth(FORMAT)
67     wf.setframerate(RATE)
68     wf.writeframes(b''.join(frames))
69     wf.close()
70
71     self.addFileToList(fileName, "waves")
72     self.syncToFile()
73
74
75
76 def addWaveFromAmp(self, fileName, freq, amp, dirName="waves/", addToList=True):
77     scipy.io.wavfile.write(Db.prefixPath + dirName + fileName, freq, int16(amp))
78     if addToList:
79         self.addFileToList(fileName, "waves")
80         self.syncToFile()
81
82
83 def addFileToList(self, fileName, dirFile=""):
84     """ Add a file to the list. Needs that the file already exists
85     Parameters :
86     @fileName : name of the file in the storage directory prefixed by
87     dirFile
88     @dirFile : prefix of the file
89     """
90     if len(dirFile) == 0:
91         dirFile = "storage"
92     if os.access(Db.prefixPath + dirFile + "/" + fileName, os.F_OK):
93         if not fileName in self.filesList:
94             self.filesList.append(fileName)
95             self.syncToFile()
96             self.addLog("L'insertion du fichier a bien ete effectuee")
97         else:
98             self.addLog("Le fichier est dejÃ dans la bibliothÃque")
99     else:
100         self.addLog("Le fichier n'existe pas")
101
102
103 def addFile(self, fileName, content, dirFile=""):
104     """ Add a file to the list and to the storage directory
105     Parameters :
106     @fileName : name of the file in the storage directory prefixed by
107     dirFile
108     @content : the content to pickle in the file
109     @dirFile : prefix of the file
110     """
111     if len(dirFile) == 0:
112         dirFile = "storage"
113     with open(Db.prefixPath + dirFile + "/" + fileName, "w") as f:
114         pickle.Pickler(f).dump(content)
115     self.addFileToList(fileName)
116
117 def deleteFileFromList(self, fileName, dirFile=""):

```

```

118 """ Remove a file from the list but does NOT remove the file from the disk
119 Parameters :
120     @fileName : name of the file in the storage directory prefixed by
121                 dirFile
122     @dirFile : prefix of the file
123 """
124 if len(dirFile) == 0:
125     dirFile = "storage"
126 if fileName in self.filesList:
127     self.filesList.remove(fileName)
128     try:
129         dirName = os.path.dirname(fileName)
130         if os.access(Db.prefixPath + dirFile + "/" + fileName, os.F_OK):
131             os.remove(Db.prefixPath + dirFile + "/" + fileName)
132             self.addLog("Le fichier a bien ete supprime")
133         else:
134             self.addLog("Le fichier n'existe pas")
135     except OSError:
136         self.addLog("La suppression a echoue")
137     try:
138         os.rmdir(Db.prefixPath + dirFile + "/" + dirName)
139         self.addLog("Le dossier a bien ete supprime")
140     except OSError:
141         pass
142     self.syncToFile()
143     self.addLog("La suppression du fichier a bien ete effectuee")
144 else:
145     self.addLog("Le fichier n'existe pas ou n'est pas gere par la base de
146                 donnees")
147
148 def syncToFile(self):
149     """ Synchronize the list of the files stored from the current attribute """
150     try:
151         with open(Db.prefixPath + Db.filesListName + ".txt", "w") as f:
152             pickle.Pickler(f).dump(self.filesList)
153     except IOError:
154         raise Exception("Le fichier n'existe pas")
155
156
157 def recursiveSync(self, dirName = "", dirIni = "storage/"):
158     """ Synchronize the list of the files stored by studying recursively the
159         current tree """
160     for f in os.listdir(Db.prefixPath + dirIni + dirName):
161         if os.path.isfile(os.path.join(Db.prefixPath + dirIni + dirName, f)):
162             self.addFileToList(os.path.join(dirName, f), dirIni)
163         else:
164             self.sync(dirName + f + "/", dirIni)
165
166 def sync(self, dirName = "", dirIni = "storage/"):
167     #Delete files that don't exist anymore in the list
168     for k, f in enumerate(self.filesList):
169         if not os.access(Db.prefixPath + dirIni + f, os.F_OK) and not os.access(Db
170             .prefixPath + "storage/" + f, os.F_OK):
171             del self.filesList[k]
172     self.addFile(Db.filesListName + ".txt", self.filesList)
173     self.recursiveSync(dirName, dirIni)
174
175 def reset(force=False):
176     """ Reset the files list """
177     msg = "etes-vous sur de vouloir reinitialiser la liste des fichiers ? (Oui =
178         0/Non = 1)"
179     if force or int(input(msg)) == 0:
180         with open(Db.prefixPath + Db.filesListName + ".txt", "w") as f:

```

```

179         c = pickle.Pickler(f)
180         c.dump([])
181         print "Reinitialisation reussie pour les fichiers"
182     reset = staticmethod(reset)
183
184
185     def syncHmm(self):
186         hmmList = self.getFile("hmmList.txt")
187         for k,f in hmmList.items():
188             if not os.access(Db.prefixPath + "hmm/" + f,os.F_OK):
189                 del hmmList[k]
190         self.addFile("hmmList.txt", hmmList)
191
192     def printFilesList(self,dirName="",printBool=True,*extRequired):
193         """ Display the files list
194         Parameters:
195             @dirName : a prefixed
196             @*extRequired : contains the extensions to display (e.g. <.wav, .txt
197                             >) """
198         filesListExt = []
199         n = 0
200         for k,f in enumerate(self.filesList):
201             #On recupère l'extension du fichier parcouru
202             a,ext = os.path.splitext(f)
203             d = os.path.dirname(f)
204             if (dirName == "" or d == dirName ) and (len(extRequired) == 0 or ext in
205                 extRequired):
206                 if printBool:
207                     print n, " - ", k, " - ", f
208                     n += 1
209                 filesListExt.append(f)
210         return filesListExt
211
212     def printDirFiles(self,dirName="storage/"):
213         """ Display the list of files in a directory
214         Parameters :
215             @dirName = "storage/" : directory to browse """
216         dirListExt = []
217         l = os.listdir(Db.prefixPath + dirName)
218         for k,f in enumerate(l):
219             #On recupère l'extension du fichier parcouru
220             print k, " - ", f
221         return l
222
223     def __str__(self):
224         print self.filesList
225
226     def addLog(self,s,fileName=""):
227         if Db.verbose:
228             print s
229             self.log += "\n" + s
230             #self.addFile("dblog" + fileName + ".txt",self.log,"logs/")
231
232     def logDump(self,fileName,log=""):
233         if log == "":
234             name = "dblog"
235             log = self.log
236         else:
237             name = "handlinglog"
238         self.addFile(name + fileName + ".txt",log,"logs/")
239
240 if __name__ == "__main__":
241     db = Db()
242     db.addFileToList("test.txt")

```

```

243 db.getFile("test.txt")
244 print(db)

```

E.4 util.py

```

1 def is2Power(N):
2     return N == np.power(2, int(math.log(N, 2)))
3
4 def get2Power(N):
5     return int(np.power(2, int(math.log(N, 2)) + 1))
6
7 def zPad(sig):
8     N = len(sig)
9     return sig + [0 for i in range(get2Power(N) - N)]
10
11 def reduc(M,N):
12     d = gcd(M, N)
13     return M/d, N/d
14
15 def pgcd(a,b):
16     # une fonction fractions.gcd(a, b) est deja implementee dans Python
17     return gcd(a, b)
18
19 def W(k,N):
20     return np.exp(-(2*np.pi*k/N)*1j)
21
22 def restreindre(sig):
23     M = float(max(abs(sig)))
24     return [float(sig[k])/M for k in range(len(sig))]
25
26 def getSin(freq, N, freqEch=44100):
27     return [np.sin(2*np.pi*freq*t/freqEch) for t in range(N)]

```

F. SpeechApp

F.1 main.js

```

1 navigator.getUserMedia = (navigator.getUserMedia ||
2                             navigator.webkitGetUserMedia ||
3                             navigator.mozGetUserMedia);
4 window.AudioContext = window.AudioContext || window.webkitAudioContext;
5 window.URL = window.URL || window.webkitURL || window.mozURL;
6
7
8 var mediaRecorder; //Object MediaRecorder
9 //var audioElement = document.getElementById('audio'); //L'object audio pour le
direct play
10 var mediaStream; //Le flux LocalMediaStream pour moz browsers
11
12 var webkitaudio_context;
13 var webkitrecorder;
14
15

```

```

16 var recording = false;
17 var nav = null; //Enregistre le type de navigateur: moz ou webkit
18
19
20 var user = "demo";
21 var hashedPass = "8b1c1c1eae6c650485e77efbc336c5bfb84ffe0b0bea65610b721762";
22 var clientDB = "demo";
23 var SERVERURL = 'localhost:8010';
24
25
26 onload = function(){
27     /* Au chargement, si l'API MediaRecorder est supporté,
28     on initialise l'entrée audio avec l'API getUserMedia */
29     if (navigator.getUserMedia){
30         if (typeof MediaRecorder === 'undefined'){
31             if (navigator.getUserMedia && window.AudioContext && window.URL){
32                 nav = 'webkit';
33                 webkitaudio_context = new AudioContext;
34
35             }
36             else{
37                 alert("Votre navigateur ne nous supporte pas :°(");
38             }
39         }
40         else{
41             nav = 'moz';
42         }
43     }
44
45     if (nav !== null){
46         navigator.getUserMedia({audio: true},
47             initRecording,
48             function(err) {
49                 console.log("The following error occurred: " + err);
50             }
51         );
52         console.log(nav + ' compatibility mode running ...');
53     }
54 };
55
56
57
58 function initRecording(localMediaStream){
59     if (nav == 'moz'){
60         mozinitRecording(localMediaStream);
61     }
62     else if (nav == 'webkit'){
63         webkitinitRecording(localMediaStream);
64     }
65 }
66
67
68 function main(){
69     /* Decide quelle action lancer lorsque le bouton est toggled */
70     var microphone = document.getElementById('microphone');
71     if (!recording){
72         try{
73             microphone.className = "wobble animated";
74             microphone.style.border = '5px solid #003173';
75             startRecord();
76             recording = true;
77             //changeLogoBG('green');
78         }
79         catch (e){
80             console.log("Recording issue\n" + e);
81         }

```



```

82     }
83     else{
84         try{
85             stopRecord();
86             recording = false;
87             //changeLogoBG('white');
88             microphone.className = "";
89             microphone.style.border = '5px solid white';
90         }
91         catch (e){
92             console.log("Recording stop issue\n" + e);
93         }
94     }
95 }
96 }
97
98
99 function startRecord(){
100     /* Lance un enregistrement */
101     navSwitch(mozstartRecorder, webkitstartRecorder);
102     console.log('recording');
103 }
104
105 function stopRecord(){
106     /* Stopper et clore un enregistrement */
107     navSwitch(mozstopRecorder, webkitstopRecorder);
108 }
109
110
111 function navSwitch(mozaction, webkitaction){
112     console.log(nav);
113     if (nav == 'moz'){
114         mozaction();
115     }
116     else if (nav == 'webkit'){
117         webkitaction();
118     }
119 }
120
121
122 function preInteract(audioBlob, blobType){
123     if (nav == 'webkit'){
124         var url = URL.createObjectURL(audioBlob);
125     }
126     else if (nav == 'moz'){
127         var url = window.URL.createObjectURL(audioBlob.data);
128     }
129     showDlLink(url);
130
131
132     //Log dans la console
133     console.log("Data available !!!");
134     console.log(audioBlob);
135
136     //Envoie Ã la console une adresse de tÃ@lÃ@chargement de l'Ã@chantillon
137     console.log(url);
138
139     //Communique les data au serveur
140     servInteract(audioBlob.data, blobType);
141
142 }
143
144 function showDlLink(url){
145     console.log(url);
146 }
147

```

```

148 //////////////////////////////////////////////////
149 //////////////////////////////////////////////////
150 //////////////////////////////////////////////////
151
152 function mozinitRecording(localMediaStream){
153     /* Initialise l'enregistrement */
154     mediaRecorder = new MediaRecorder(localMediaStream);
155     mediaRecorder.ondataavailable = mozmediaOnDataAvailable;
156     mediaStream = localMediaStream;
157
158     console.log('getUserMedia initialised');
159 }
160
161
162 function mozstartRecorder(){
163     mediaRecorder.start();
164     //var audioElement = document.getElementById('audio');
165     //audioElement.src = window.URL.createObjectURL(mediaStream);
166     //console.log(audioElement.src);
167 }
168
169
170 function mozstopRecorder(){
171     if (mediaStream){
172         console.log('stopRecord');
173         mediaRecorder.stop();
174         //var audioElement = document.getElementById('audio');
175         //audioElement.src = '';
176     }
177 }
178
179
180 function mozmediaOnDataAvailable(blob){
181     /* A la fin de l'enregistrement, r  cup  re le blob dans data
182        et lance le traitement */
183     console.log("moz data available");
184     preInteract(blob, 'ogg');
185 }
186
187 //////////////////////////////////////////////////
188 //////////////////////////////////////////////////
189
190 function webkitinitRecording(localMediaStream){
191     var input = webkitaudio_context.createMediaStreamSource(localMediaStream);
192     //input.connect(webkitaudio_context.destination);
193     webkitrecorder = new Recorder(input);
194
195 }
196
197
198 function webkitstartRecorder(){
199     webkitrecorder && webkitrecorder.record();
200 }
201
202
203 function webkitstopRecorder(){
204     webkitrecorder && webkitrecorder.stop();
205
206     webkitrecorder && webkitrecorder.exportWAV(function(audioBlob) {
207         preInteract(audioBlob, 'wav');
208     });
209
210     webkitrecorder.clear();
211 }
212
213 //////////////////////////////////////////////////

```

```

214 //////////////////////////////////////////////////
215 //////////////////////////////////////////////////
216
217 function servInteract(audioBlob, blobType){
218     // Envoie le blob au serveur
219     var formData = new FormData();
220     formData.append('user', user);
221     formData.append('hashedPass', hashedPass);
222     formData.append('clientDB', clientDB);
223
224     formData.append('action', 'recognize_spoken_word');
225
226     formData.append('audioBlob', audioBlob);
227     formData.append('audioType', blobType);
228
229     var req = new XMLHttpRequest();
230     req.open('POST', 'handler', false);
231     /*req.onstatechange = function(){
232         console.log('ez');
233         console.log(req.readyState);
234         if (req.readyState === 4){
235             console.log('4');
236             if (req.status === 200){
237                 console.log(200);
238                 wordResponse(req.responseXML);
239             }
240         }
241     }*/
242     //req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
243     req.send(formData);
244     console.log(req);
245     resp = req.responseXML;
246     console.log(resp);
247     wordResponse(resp);
248
249 }
250
251
252 function wordResponse(respXML){
253     if (respXML.getElementsByTagName('respWord')){
254         var responseWord = respXML.getElementsByTagName('respWord')[0].textContent;
255         console.log(responseWord.name);
256     }
257     else{
258         var responseWord = "Error : '(";
259     }
260     var responseElement = document.getElementById('responseWord');
261     console.log(responseWord);
262     responseElement.innerHTML = responseWord;
263
264 }

```

F.2 index.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3     <head>
4         <meta charset="utf-8">
5         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6         <meta name="viewport" content="width=device-width, initial-scale=1.0">
7         <meta name="description" content="MIG SE Speech Recognition Demonstrator">
8         <meta name="author" content="MIG SE Team">
9         <link rel="shortcut icon" href="img/favicon.png">

```

```

10 <title>SpeechApp demonstrator</title>
11
12
13 <!-- Bootstrap core CSS -->
14 <link href="css/bootstrap.min.css" rel="stylesheet">
15 <!-- Bootstrap theme -->
16 <link href="css/bootstrap-theme.min.css" rel="stylesheet">
17
18 <!-- Custom styles for this template -->
19 <link href="css/theme.css" rel="stylesheet">
20
21 <!-- Just for debugging purposes. Don't actually copy this line! -->
22 <!--[if lt IE 9]><script src="../../docs-assets/js/ie8-responsive-file-warning.js
    "></script><![endif]-->
23
24 <!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries --
    >
25 <!--[if lt IE 9]>
26 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script
    >
27 <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></
    script>
28 <![endif]-->
29 </head>
30
31 <body>
32
33 <!-- Fixed navbar -->
34 <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
35 <div class="container">
36 <div class="navbar-header">
37 <button type="button" class="navbar-toggle" data-toggle="collapse" data-
    target=".navbar-collapse">
38 <span class="sr-only">Toggle navigation</span>
39 <span class="icon-bar"></span>
40 <span class="icon-bar"></span>
41 <span class="icon-bar"></span>
42 </button>
43 <a class="navbar-brand" href="#">SpeechApp</a>
44 </div>
45 <div class="navbar-collapse collapse">
46 <ul class="nav navbar-nav">
47 <li class="active"><a href="#">Home</a></li>
48 <li><a href="#services">Our services</a></li>
49 <li class="dropdown">
50 <a href="#" class="dropdown-toggle" data-toggle="dropdown">About <b
    class="caret"></b></a>
51 <ul class="dropdown-menu">
52 <li><a href="#team">Our team</a></li>
53 <li><a href="#work">Our work</a></li>
54 <li><a href="#contact">Contact</a></li>
55 </ul>
56 </li>
57 </ul>
58 </div><!--/.nav-collapse -->
59 </div>
60 </div>
61
62 <div class="container theme-showcase">
63
64 <!-- Main jumbotron for a primary marketing message or call to action -->
65 <div class="jumbotron">
66 <h1>SpeechApp !</h1>
67 <p>Welcome to this demonstrator of our state of the art speech recognition
    technology</p>
68 <p>Test it, feed us with a few words</p>

```

```

69     <p><a href="#services" class="btn btn-primary btn-lg" role="button">Learn
70         more &raquo;</a></p>
71 </div>
72
73
74 <div class="page-header" id="demonstrator">
75     <h1>Demonstrator</h1>
76 </div>
77
78 <p>
79     <div style="text-align: center;" id="sound-recording">
80         
82
83         <p id="recording">
84             <audio autoplay src="" id="audio">audio</audio>
85         </p>
86         <p id="responseWord" style="color: blue" ></p>
87     </div>
88 </p>
89
90 <div class="page-header" id="services">
91     <h1>Our services</h1>
92 </div>
93
94 <p>
95
96 </p>
97
98 <div class="page-header" id="team">
99     <h1>Our team</h1>
100 </div>
101
102 <p>
103     <p style="text-align: center;">
104         
105     </p>
106     We are a team of 13 first year students at <a href="http://www.mines-
107         paristech.eu/">MINES ParisTech</a><br />
108 </p>
109 <div class="page-header" id="work">
110     <h1>Our work</h1>
111 </div>
112
113 <p>
114     We've been working for 3 weeks on speech recognition at the <a href="http://
115         www.cma.ensmp.fr/">CMA</a>. We've build from scratch an isolated
116     spoken word recognition engine that works pretty well, and the tools to use
117     it easily. This webapp is
118 </p>
119 <p>Our project has been written with Python, and its source is hosted on
120     <a href="https://github.com/giliam/mig2013">our Git repository</a></p>
121
122 <div class="page-header" id="contact">
123     <h1>Contact</h1>
124 </div>
125
126 <p>
127     You'll find our GitHub profiles on
128     <a href="https://github.com/giliam/mig2013">our Git repository page</a>.<br />
129     >
130     The maintainer of that webapp can be contacted at

```

```

129     <a href="mailto:speechapp@wumzi.info">speechapp@wumzi.info</a>.
130 </p>
131
132
133 </div> <!-- /container -->
134
135 <!-- SpeechApp js core -->
136 <script src="js/main.js"></script>
137 <script src="js/recorder.js"></script>
138
139
140 <!-- Bootstrap core JavaScript
141 ===== -->
142 <!-- Placed at the end of the document so the pages load faster -->
143 <script src="js/jquery-1.10.2.min.js"></script>
144 <script src="js/bootstrap.min.js"></script>
145 <script src="js/holder.js"></script>
146 </body>
147 </html>

```

G. SpeechServer

G.1 main.py

```

1 class SpeechServerHandler( BaseHTTPServer.BaseHTTPRequestHandler ):
2     def do_GET(self):
3         """ Respond to a GET request """
4         self.send_response(200)
5         self.send_header('Content-type', 'text/plain')
6         self.end_headers()
7         self.wfile.write("Ca se passe en POST pour les requetes !")
8
9
10    def do_POST(self):
11        """Respond to a POST request"""
12
13        form = FieldStorage(
14            fp=self.rfile,
15            headers=self.headers,
16            environ={'REQUEST_METHOD': 'POST',
17                    'CONTENT_TYPE': self.headers['Content-Type'],
18                    })
19
20        user = form.getvalue('user')
21        hashedPass = form.getvalue('hashedPass')
22        clientDB = '#form.getvalue('clientDB')
23
24        #form = dict(form)
25        #print(form)
26
27        #Check if the user is authorized and he has access to clientDb
28        authUser = AuthUser()
29        if True or authUser.checkAuth(user, authUser.hashPass(hashedPass), clientDB):
30            action = form.getvalue('action')
31            requestHandler = requestHandling()
32            respData = requestHandler.handle(clientDB, action, form)

```

```

33         respXML = self.buildXMLResponse(respData)
34     else:
35         respXML = "You're not authorized to call me !\
36                 Register at speech.wumzi.info"
37
38     #respXML = self.buildXMLResponse({'respWord' : user})
39
40
41     #And respond
42     self.send_response(200)
43     self.send_header('Content-type', 'text/xml')
44     self.end_headers()
45     self.wfile.write(respXML)
46
47     @classmethod
48     def buildXMLResponse(cls, data):
49         """Build the XML doc response from the data dictionnary"""
50         root = ET.Element('root')
51         for key, value in data.items():
52             elem = ET.SubElement(root, key)
53             elem.text = value
54
55         return ET.tostring(root, encoding="utf-8")
56
57
58     @classmethod
59     def parseXMLRequest(cls, XMLString):
60         """Build a dict from an XML doc"""
61         data = {}
62         root = ET.fromstring(XMLString)
63         for child in root:
64             data[child.tag] = child.text
65         return data
66
67
68
69 def run(port, adress="localhost"):
70     server = BaseHTTPServer.HTTPServer((adress, port), SpeechServerHandler)
71     server.serve_forever()
72
73
74 if __name__ == '__main__':
75     import sys
76     if len(sys.argv) >= 2:
77         try:
78             PORT = int(sys.argv[1])
79         except TypeError:
80             print("Please provide an int !")
81     else:
82         PORT = 8010
83         print("Port set to default : %s" % PORT)
84
85     run('localhost', PORT)

```

G.2 audioConverter.py

```

1  TMP_DIR = "tmp/"
2
3  def path_orig_ogg(id):
4      return TMP_DIR + "orig_" + str(id) + ".ogg"
5
6
7  def path_mid_wave(id):

```

```

8     return TMP_DIR + "orig_" + str(id) + ".wav"
9
10 def path_mid_wave_split(id):
11     return TMP_DIR, "orig_" + str(id) + ".wav"
12
13 def path_final_wave(id):
14     return TMP_DIR + "wave_final_" + str(id) + ".wav"
15
16 def path_final_wave_split(id):
17     return TMP_DIR, "wave_final_" + str(id) + ".wav"
18
19 def rm_multi(*files):
20     """Remove multiple files"""
21     for path in files:
22         os.remove(path)
23
24 def handleOGGBlob(oggBlob):
25     """Converti le blob ogg en blob wav"""
26     id = randint(1, 1000)
27     while os.access(path_orig_ogg(id), os.W_OK):
28         id = randint(1, 1000)
29
30     writeBlobToDisk(oggBlob, path_orig_ogg(id))
31
32     #Now convert the file
33
34     print(path_orig_ogg(id))
35     os.system('soundconverter -b -m audio/x-wav -s .wav "%s"' % path_orig_ogg(id))
36     #Resample to 44.1kHz
37
38     return finalHandling(id)
39
40
41 def finalHandling(id):
42     print('final handling started')
43     os.system('sox -r 44.1k -e signed -c 1 -b 16 %s %s' % (path_mid_wave(id),
44         path_final_wave(id)))
45     print('soxed')
46     #And read the oggBlob
47
48     '''with open(path_final_wave(id), 'r') as finalwavefile:
49         waveBlob = finalwavefile.read()'''
50
51     dir, file = path_final_wave_split(id)
52     print(dir, file)
53     waveBlob = cutsyncaudio(dir, file)
54
55     #Remove the files
56     rm_multi(path_mid_wave(id), )#path_final_wave(id))
57     print("success")
58     return waveBlob
59
60 def handleWAVBlob(audioBlob):
61     id = randint(1, 1000)
62     while os.access(path_mid_wave(id), os.W_OK):
63         id = randint(1, 1000)
64
65     writeBlobToDisk(audioBlob, path_mid_wave(id))
66     print(path_mid_wave(id))
67     print("bringing id to finalHandling")
68     return finalHandling(id)
69
70
71
72 def writeBlobToDisk(audioBlob, path):

```



```

73     with open(path, 'w') as origfile:
74         origfile.write(audioBlob)
75         origfile.flush()
76         os.fsync(origfile)
77
78
79 def cutsyncaudio(dir, file):
80     sync.cutBeginning(dir, file, prefix='')
81     sync.syncFile(dir, file, prefix='')
82
83     waveBlob = scipy.io.wavfile.read(dir + file)
84
85     return waveBlob
86
87
88 def convert_ogg_to_wav(ogg_path, out_wav_path):
89     try:
90         with open(ogg_path, 'r') as origoggfile:
91             waveBlob = handleOGGBlob(origoggfile.read())
92     except:
93         return "Impossible to open ogg file"
94
95     try:
96         with open(out_wav_path, 'w') as out_wav:
97             out_wav.write(waveBlob)
98             out_wav.flush()
99             os.fsync(out_wav)
100    except:
101        return "Impossible to write wav blob to dest file"
102
103    return waveBlob
104
105 def sox_handling(wavBlob, pathToTmp="../db/waves/tmp/"):
106     tempFileName = hashlib.sha224(str(randint(0,1e10))).hexdigest()
107     fileName = pathToTmp + str(tempFileName) + ".wav"
108     with open(fileName, 'w') as origoggfile:
109         origoggfile.write(wavBlob)
110         origoggfile.flush()
111         os.fsync(origoggfile)
112
113     os.system('ffmpeg -i "' + fileName + '" -vn -ss 00:00:00 -t 00:00:01 "' +
114             pathToTmp + 'noiseaud.wav"')
115     print("noise extracted")
116     os.system('sox "' + pathToTmp + 'noiseaud.wav" -n noiseprof "' + pathToTmp + '
117             noise.prof"')
118     print("noise selected")
119     sleep(1)
120     os.system('sox "' + fileName + '" "' + fileName + '" noisered "' + pathToTmp + '
121             noise.prof" 0.21')
122     print("noise trashed")
123     #os.remove(pathToTmp + "noise.prof")
124     #os.remove(pathToTmp + "noiseaud.wav")
125
126     wav_content = scipy.io.wavfile.read(fileName)
127     return wav_content
128
129 if __name__ == '__main__':
130     print(sox_handling(convert_ogg_to_wav('test.oga', 'test.wav')))
```

G.3 clientAuth.py

```
1 DEBUG = False
```

```

2
3 class AuthUser:
4     """ Classe pour gérer l'authentification des applications clientes sur notre
5         serveur applicatif """
6
7     def __init__(self, fileName="registre"):
8         self.userListFile = fileName
9         self.db = Db("../db/", "userDbList", DEBUG)
10        self.userList = self.db.getFile("users/" + fileName + ".txt")
11        self.username = ""
12        self.password = ""
13        self.connected = False
14
15
16    def newClient(self, client, hashedPass, authorizedDBs):
17        """ Ajoute un utilisateur et des données """
18        if not self.getClient(client):
19            self.userList[client] = hashedPass, authorizedDBs
20            self.commit()
21            return True
22        return False
23
24    def updateClient(self, client, hashedPass, authorizedDBs):
25        """ Ajoute un utilisateur et des données """
26        if self.userList.get(client):
27            self.userList[client] = [hashedPass, authorizedDBs]
28            self.commit()
29            return True
30        return False
31
32
33    def rmClient(self, client):
34        """ Supprime un client du dictionnaire """
35        if self.userList.get(client):
36            del self.userList[client]
37            self.commit()
38            return True
39        return False
40
41
42    def getClients(self):
43        print self.userList
44
45    def getClient(self, client):
46        """ Retourne un client s'il se trouve dans la liste des utilisateurs """
47        if self.userList.get(client):
48            return self.userList[client]
49        else:
50            return False
51
52
53    def checkAuth(self, client, submittedHashedPass, clientDB=""):
54        """ Vérifie que le nom entré se trouve bien dans la liste des utilisateurs
55            """
56        if self.getClient(client):
57            hashedPass, clientDBs = self.getClient(client)
58            if hashedPass == submittedHashedPass:
59                if clientDB == "" or clientDB in clientDBs:
60                    return True
61            return False
62
63    def login(self, client, submittedHashedPass):
64        """ Connecte l'utilisateur """
65        if self.getClient(client):
66            hashedPass, clientDBs = self.getClient(client)
67            if hashedPass == submittedHashedPass:

```

```

67         self.username = client
68         self.password = submittedHashedPass
69         self.connected = True
70         return True
71     return False
72
73     def logOut(self):
74         self.username = ""
75         self.password = ""
76         self.connected = False
77
78     def hashPass(self, password):
79         return hashlib.sha224(password).hexdigest()
80
81
82     def commit(self):
83         """ Write the changes of the userlist to the DB on disk """
84         self.db.addFile("users/" + self.userListFile + ".txt", self.userList)
85
86
87     def __str__(self):
88         """ Affiche la liste des utilisateurs et leurs donnÃ©es """
89         data = ["%s :\t %s" % (client, clientData) for client, clientData in self.
90                 userList.items()]
91         return '\n'.join(data)
92
93 if __name__ == "__main__":
94     authUserHandler = AuthUser()
95     print(authUserHandler)
96     authUserHandler.newClient("demo", "demo", [1])
97     print(authUserHandler)

```

G.4 speechActions.py

```

1 ACTIONS = ["add_word", "list_word_records", "rm_word_record", "recognize_spoken_word", "
2     listen_recording"]
3
4 class requestHandling:
5     def handle(self, clientDBid, action, data):
6         self.dbWaves = Db('../db/', verbose=False)
7
8         if not action in ACTIONS:
9             return False
10
11         if action == "recognize_spoken_word":
12             audioBlob = data.getvalue("audioBlob")
13             audioType = data.getvalue("audioType")
14             if not (audioBlob or audioType):
15                 return None
16             else:
17                 return self.recognize_spoken_word(audioBlob, audioType, clientDBid)
18
19     def recognize_spoken_word(self, audioBlob, audioType, clientDBid):
20         """ Handle the specific request to recognize a spoken word """
21         TYPES = ['wav', 'ogg']
22         if audioType not in TYPES:
23             return False
24
25         if audioType == 'ogg':
26             audioBlob = handleOGGBlob(audioBlob)
27             print(audioBlob)
28         elif audioType == 'wav':

```

```
28         audioBlob = handleWAVBlob(audioBlob)
29
30         #wav_content = sox_handling(audioBlob)
31         #print wav_content
32         respWord, log = handlingOneWord(audioBlob[1], self.dbWaves, 1, 1, 0)
33         print respWord
34         return {'respWord': respWord}
```

Bibliographie

- [1] Apple. Application siri. <http://www.apple.com/fr/ios/siri/>, 2013.
- [2] Tom Preston-Werner, Chris Wanstrath, and PJ Hyett. Github. <https://github.com/giliam/mig2013/>, 2013.
- [3] Lawrence Rabiner. *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1993.
- [4] MIT. Pyaudio. <http://people.csail.mit.edu/hubert/pyaudio/>, 2006.
- [5] Stanley Smith Stevens, John Volkman, and Edwin B. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 1937.
- [6] Begam, Elamvazuthi, and Muda. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw). *Journal of Computing*, 2010.
- [7] Vincent Arsigny. Modélisation par un champ de markov du signal de parole et application à la reconnaissance vocale. Technical report, École Nationale Supérieure des Télécom de Paris, 2000.
- [8] Zohar Babin. How to do noise reduction using ffmpeg and sox. <http://www.zoharbabin.com/how-to-do-noise-reduction-using-ffmpeg-and-sox/>, 2011.
- [9] Chris Bagwell. Sox website. <http://sox.sourceforge.net/Docs/Documentation>, 2009.
- [10] H.G. Hirsch, P Meyer, and H.W. Ruehl. Improved speech recognition using high-pass filtering of subband envelopes. *Eurospeech*, 1991.
- [11] Anonyme. Théorème de nyquist-shannon. http://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_d%27%C3%A9chantillonnage_de_Nyquist-Shannon, 2013.
- [12] Luc Maranget. *Introduction à la programmation*. École Polytechnique, 2008-2009.
- [13] *An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology*, 1966.
- [14] Maurice Charbit. Reconnaissance de mots isolés (utilisation des modèles HMM). *Inconnu*, Oct. 2002.
- [15] Franck Bonnet, Benjamin Devèze, Mathieu Fouquin, and Julien Jeany. Reconnaissance automatique de la parole. *Epita*, 2004.
- [16] Nuance Company. Dragon naturallyspeaking. <http://www.nuance.com/dragon/index.htm>, ?
- [17] Aharon Etengoff. Nuance clinches speech-recognition deal with us army. <http://www.itexaminer.com/nuance-clinches-speech-recognition-deal-with-us-army.aspx>, 2009.
- [18] Lockwood Reed. The requirements and applications of speech recognition technology for voice activated command and control in the tactical military environment. <http://www.dtic.mil/dtic/tr/fulltext/u2/a465047.pdf>, 2004.
- [19] Thanassis Trikas. Automated speech recognition in air traffic control. *MIT*, 1987.
- [20] G2 Speech. La reconnaissance vocale pour hôpitaux et autres institutions de soins. <http://www.g2speech.be/reconnaissance-vocale.html>, 2000.
- [21] Yves Drothier. La reconnaissance vocale au service des médecins du chu de rouen. *JDN Solutions*, 2006.
- [22] Nuance Company. Dragon naturallyspeaking for law enforcement. <http://www.nuance.com/naturallyspeaking/industries/law-enforcement/>, ?

- [23] Nancy Manasse. Speech recognition. *University of Nebraska-Lincoln*, 1990.
- [24] PrismaMedia. Capital. www.capital.fr, ?
- [25] Michael Page. Hays, officeteam, 2009-2010.
- [26] Gouvernement français. site des impôts. www.impots.gouv.fr, 2013.