MINES ParisTech

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

IL FAUT REMERCIER LES INTERVENANTS INDUSTRIELS ET AUTRES

Table des matières

Introduction

Le MIG Systèmes-Embarqués nous a demandé de nous attaquer au sujet complexe et pluridisciplinaire du « traitement automatique de la parole ».

Ce document décrit comment, 13 élèves-ingénieurs en première année à MINES ParisTech et ne possédant pas de compétences a priori sur ce domaine, se sont approprié ce sujet et ont réussi à le résoudre à travers la réalisation, de a à z, d'un logiciel de reconnaissance que nous avons baptisé SpeechApp.

Ce logiciel est visible à l'URL ? ? ?.

0.1 Présentation des enjeux

La reconnaissance vocale automatisée est l'objet d'intenses recherches depuis plus de 50 ans. Malgré son caractère d'abord futuriste, comme cela peut se retrouver dans de nombreuses œuvres de science-fiction, elle a pris sa place dans notre quotidien avec la prolifération de systèmes qui embarquent une telle technologie, par exemple avec 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, ...). Les derniers 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 croissante puissance de calcul disponible et les dernières avancées et applications découvertes.

La complexité de ce problème s'explique notamment par la grande diversité des thèmes qui lui sont connexes et que tout système se voulant performant se doit d'incorporer (traitement du signal, théorie de l'information, acoustique, linguistique, intelligence artificielle, physiologie, psychologie, ...). La reconnaissance vocale requiert des connaissances trop diverses pour être maîtrisées par un seul individu et la capacité à savoir exploiter des ressources dont on est pas expert devient un atout capital. Elle ne se réduit pas à la seule détermination d'une suite de mots prononcés, mais peut s'étendre à diverses autres applications telles que la détermination du langage, de l'accent, du sexe, de l'âge, de l'état (stressé ou calme), ou même de l'environnement du locuteur, tant ces paramètres influent de manière capitale sur l'analyse.

0.2 Objectifs du projet

Ce projet de 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 recherches et de développement ; mais plutôt, à l'instar de l'ingénieur généraliste, de prendre connaissance d'un sujet et d'une problématique et tâcher, en équipe, d'y apporter une solution qui soit la plus optimale possible compte tenu des exigences 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 tel qu'on l'expérimente aujourd'hui, n'est en fait pas si évident qu'il y paraît. En témoigne la faible réussite de ces applications

en général puisque nous avons tous ressenti un jour la frustration de ne pas être compris de la machine. Il convient donc de préciser ce qui rend la tâche si subtile face à ce que nos oreilles et notre cerveau font aussi instantanément.

Les rôles de chacun dans l'équipe de MIG ont été attribués dès le début selon les goûts et compétences de chacun mais la pertinente répartition des tâches, la diversité intrinsèque au projet et l'angle avec lequel nous l'avons abordé ont permis à chacun d'exploiter un panel très diversifié de ses compétences tout en apportant la valeur ajoutée de sa spécialité. Chaque fonction dépendant très fortement de ce qui précède et de ce qui suit, une bonne communication interne était indispensable pour un développement juste et efficace. Si la coordination spontanée d'une équipe de treize personnes a été au début délicate, une indéniable rigueur et discipline adjointe à l'exploitation de ressources adaptées ont vite imposé une organisation naturelle. Par exemple l'utilisation de la plateforme github[2] de gestion de l'échange et des mises à jour des fichiers s'est révélée particulièrement efficace et permettait à chacun d'incorporer en temps réels ses dernières modifications et ses derniers apports. La complexité de la discipline de reconnaissance de la parole 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. Le caractère abscons de certains articles a rajouté à la difficulté.

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'attaques du problème donné pouvant être considérés, ainsi que de présenter celui que nous avons choisi, avec quelles motivations.

Dans son livre *Fundamentals of speech recognition*, Lawrence Rabiner[3] dégage des travaux de ces prédécesseurs trois approches conceptuelles du problème. Ces approches sont les suivantes : 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 choisi 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 a 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 [1]. Ceci dans le but d'identifier une suite de phonèmes et d'ainsi reconnaître un mot.

Cette approche suppose qu'il existe un ensemble fini de phonèmes différentiables et que leurs propriétés sont suffisamment manifestes pour être extraites d'un signal ou de la donnée de son spectre [2] au cours du temps. Même si il est évident que ces caractéristiques dépendent très largement du locuteur, on part du principe que les règles régissant la modification des paramètres peuvent être apprises et appliquées.

1. La définition wikipédia d'un phonème est la suivante : 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.

2. Données des fréquences et de leur amplitude associée, composant un signal à un instant donné

Bien qu'elle 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 de ce 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 préalablement carractérisés, afin de le classifier en fonction du degré de concordance.

Sans entrer dans plus 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 son environnement
— Elle donne lieu à de très bons résultats

Première partie

Démarche Technique

1. Principe général du traitement de la parole

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 prononcé par un locuteur 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. L'idée est de désigner un mot par un certain motif puis de permettre par le même procédé appliqué sur un enregistrement quelconque, de parvenir à identifier deux motifs proches qui correspondraient vraissemblablement au même mot. Il s'agit donc tout d'abord de traiter le signal représentant le son 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 pratiquement impossible une simple identification par comparaisons temporelles.

1.2 Schéma global de la reconnaissance de la parole

Afin de gérer les difficultés inhérentes à la caractérisation des sons, nous avons mis en place plusieurs étapes de traitement de manière à obtenir cette fameuse « trace » qui caractériserait un enregistrement, c'est-à-dire un mot. Nous avons pour cela eu recours à plusieurs techniques de traitement du signal [1] classiquement utilisées, comme *l'échantillonnage* (capture des valeurs d'un signal à des intervalles de temps réguliers, afin, par exemple, d'en avoir une représentation en un format digital compréhensible par un ordinateur), le *fenêtrage* (découpage temporel d'un signal en petits intervalles de temps), la *transformée de Fourier directe et inverse* (permet le passage et le passage inverse : de la représentation d'un son par des amplitudes des signaux en fonction du temps à la représentation d'un son par des amplitudes des signaux en fonction des fréquences [2] qui le composent). Cette figure explique 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 pourrons manipuler en le sachant représentatif et caractéristique du son.

Enregistrement du son (de la parole au son) La première étape consiste simplement à enregistrer le son sur le disque dur de l'ordinateur. 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 microphone.

1. D'après Wikipédia, le traitement du signal est la discipline qui développe et étudie les techniques de traitement, d'analyse et d'interprétation des signaux.
2. La fréquence peut être assimilée à la hauteur d'un son.

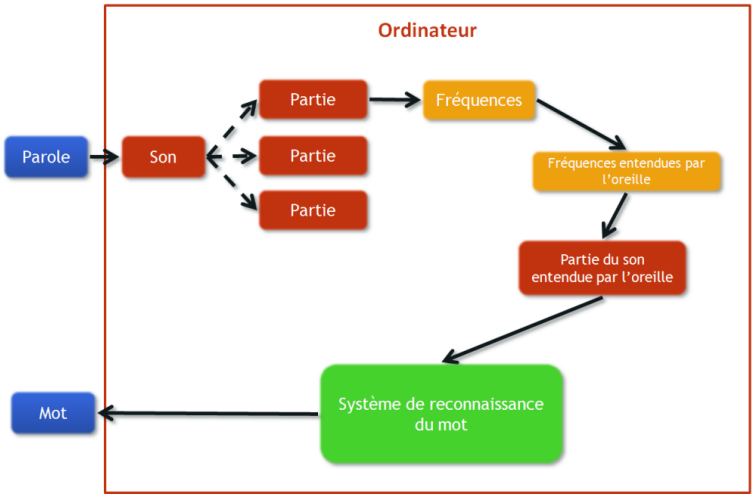

FIGURE 1.1 – Traitement du son pour le reconnaître

Découpage en fenêtre (du son en parties de son) 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 fréquences Jusque là, le son étudié se 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.

Fréquences entendues par l'oreille 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 (appelée échelle de Mel) qui accentue certaines fréquences. En effet, il a été montré[4] (et ensuite appliqué [5]) que l'oreille ne perçoit pas toutes les fréquences de la même façon.

Système de reconnaissance du mot Les traitements précédents (enregistrement, fenêtrage, passage en fréquences, filtrage par une échelle de Mel) permettent d'extraire un ensemble des valeurs qui vont caractériser un son.

L'idée sous-jacente au système informatique de reconnaissance de la parole est de construire pour chacun des mots que l'on veut reconnaître un modèle de ce mot fondé sur ces valeurs, ce modèle sera constitué des suites des sons.

Comme ces caractéristiques varient quand un même locuteur prononce le même mot plusieurs fois, il est important que le modèle de chacun des mots prenne en compte ces variations.

Ces raisons nous ont conduit à retenir pour la modélisation des *suites temporelles des caractéristiques des sons qui constituent un mot* les modèles probabilistes de Markov cachés qui, comme nous le verrons par la suite, sont tout particulièrement adaptés à ce problème. **2.**

Analyse de la parole

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 apte à l'analyse automatisée. Le principe est simple : garder les traits distinctifs du signal et éviter au maximum de 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 ?

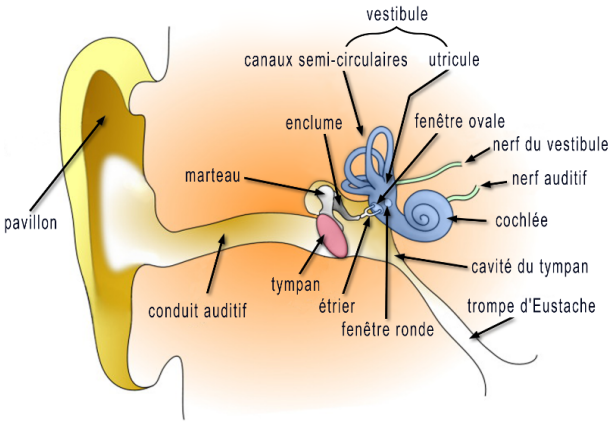

FIGURE 2.1 – Oreille humaine

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 [6]). 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, on définit aisément le *bitrate*, qui correspond au débit d'information par seconde, de la façon suivante : $bitrate = n * échantillonnage$. Ce dont nous disposons donc pour analyser un signal, est la donnée de l'amplitude en fonction du temps la caractérisant.

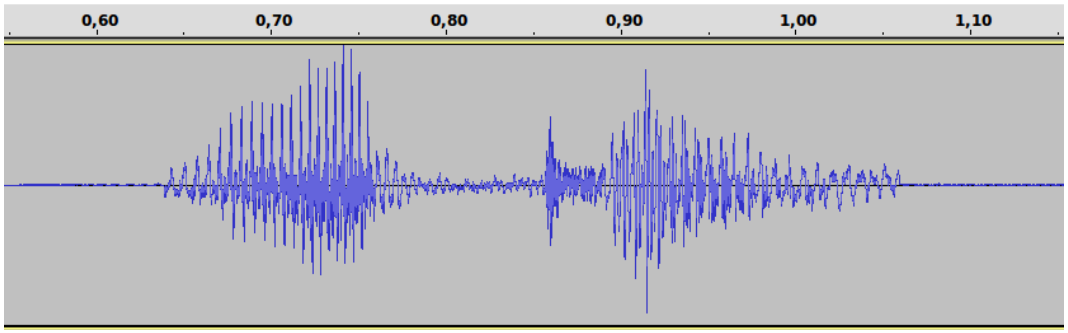

FIGURE 2.2 – Exemple audiogramme prononciation du mot "VICA"

2.3 Enregistrement, recadrage, découpage et filtrage des signaux

2.3.1 Synchronisation des enregistrements

Afin de synchroniser le début des enregistrements d'un mot, et de leur donner la même durée, il a été nécessaire 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 définitevement le problème, l'utilisateur ne pouvant pas toujours se placer dans ces conditions, le signal est traité par un filtre anti-bruit.

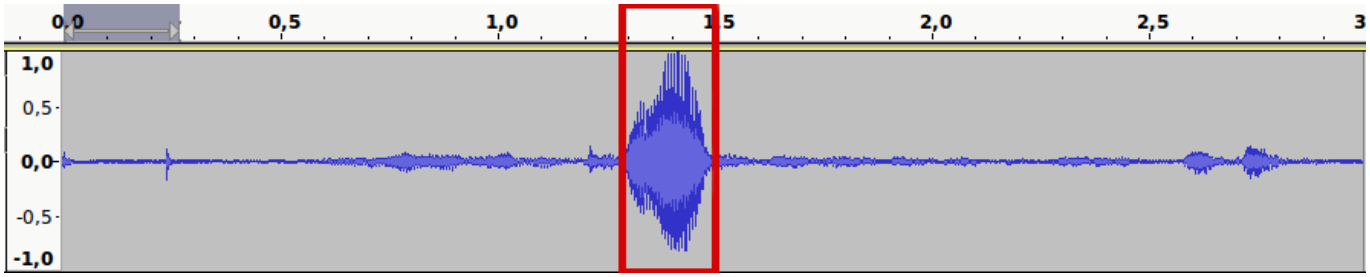

FIGURE 2.3 – Ensemble du son enregistré, la partie nous intéressant (le mot) est encadrée en rouge

Ce filtre consiste en l'utilisation de bibliothèques, SoX et ffmpeg([7] et [8]), 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 complétement 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 unes 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 Filtrage des signaux

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 immanquablement un filre passe haut c'est-à-dire une accentuation de l'amplitude associée 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 propose 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.
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.

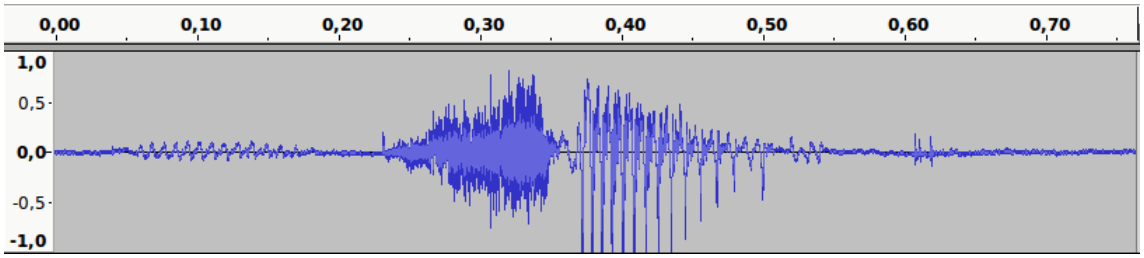

FIGURE 2.4 – Exemple d'un fichier son (représentant « Cinq ») **avant** application du filtre

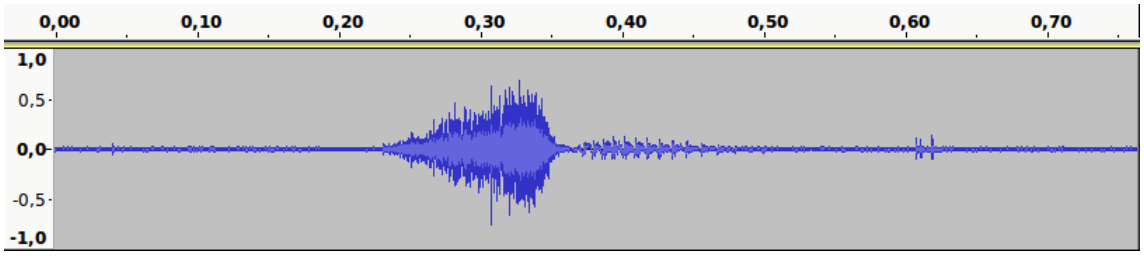

FIGURE 2.5 – Exemple du fichier son **après** application du filtre

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.

2.4 Echantillonnage et fenêtrage des signaux

L'analyse du signal, pour accéder au domaine fréquentiel, s'affranchit de la dépendance temporelle. Le spectre obtenu ne correspond plus à une perception physique, mais à une moyenne temporelle du spectre perçu. 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 le plus proche possible de la continuité. Il est en revanche nécessaire de conserver un certain nombre de points par échantillons. En effet, le spectre obtenu par l'analyse sera plus précis et proche de la réalité fréquentielle si le nombre de point 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[9] 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 logiciel à notre disposition se cantonnait 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 évènements sonores et variations s'étalant sur une durée inférieure à 20 millisecondes n'étaient pas signifiants pour notre analyse. Le nombre de points a été par cette donnée, couplée à notre fréquence d'échantillonnage lors des enregistrements, à 44100Hz.

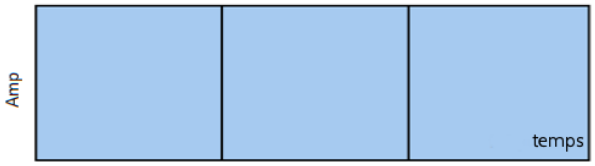

FIGURE 2.6 – Principe normal du fenêtrage

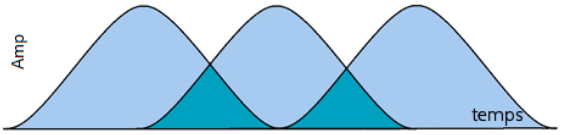

FIGURE 2.7 – Fenêtre de Hann évitant les discontinuités

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 à l'autre), au prix d'une multiplication du temps de calcul par deux.

2.5 Passage des signaux dans le domaine des fréquences

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.

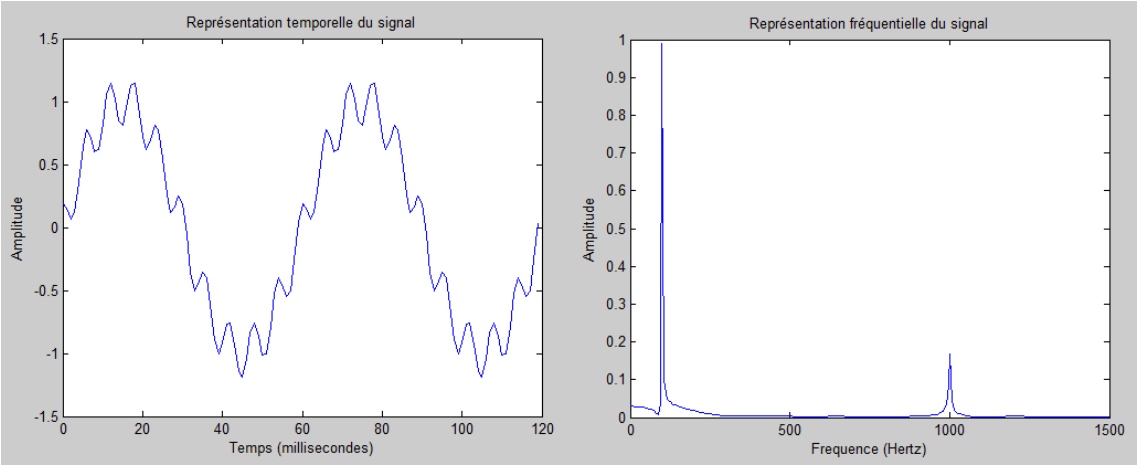

FIGURE 2.8 – Exemple de passage du domaine temporel (somme de cosinus) au domaine fréquentiel (pics pour les fréquences fondamentales)

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. Notre fonction se base sur l'algorithme de Cooley-Tukey, qui permet de réduire la complexité à $O(Nlog2(N))$, 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 ce 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 eu 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 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 appliqué est de passer le code de Python à 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. 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 :

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

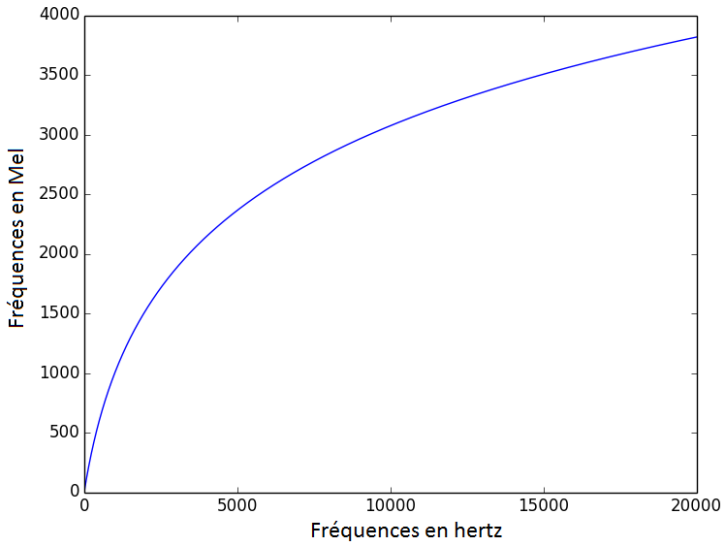

FIGURE 2.9 – Graphe de conversion

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.

2.7 Passage inverse des signaux dans le domaine temporel

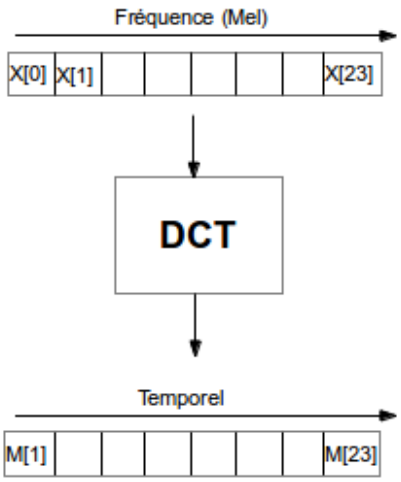

FIGURE 2.10 – Graphe de conversion

Dans les parties précédentes nous avons vu comment, à partir d'un extrait sonore échantillonné à 44100 Hz sur 16 bits, obtenir après transfrormée 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 préfèrera une représentation en temporel de la fraction sonore considérée, ceci pour deux principales raisons :

Opérer une transformation en cosinus inverse « décorelle » les valeurs du tableau dans la mesure où dans une représentation en fréquentiel, les valeurs associées aux hautes fréquences sont très fortement corellées avec ce qui se passe dans les basses fréquences. En effet, un signal sonore n'est jamais pur, c'est-à-dire contitué 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 :

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

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

3. Modélisation des mots par modèles de Markov cachés (MMC)

3.1 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[10]. 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.

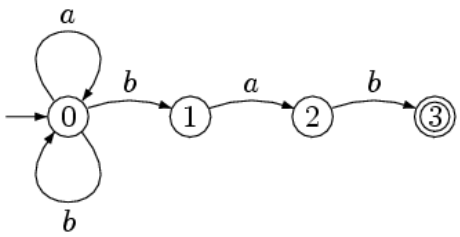

FIGURE 3.1 – Exemple d'automate « classique »

Les modèles de Markov cachés sont largement répandus dans la reconnaissance vocale([11], [3] et [12]). 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.

3.2 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.

3.3 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".

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.

3.4 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 accroit 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 avions des besoins matériels (stocker l'ensemble des mots reconnus) mais aussi des besoins humains, et en l'occurrence une diversité de voix.

3.5 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.

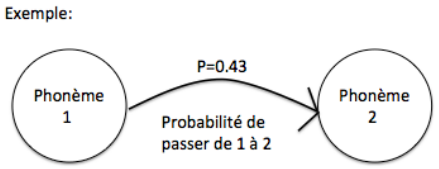

FIGURE 3.2 – Exemple de deux phonèmes et de la probabilité de passer du phonème 1 au phonème 2

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 50 000. 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. # Récapitulatif

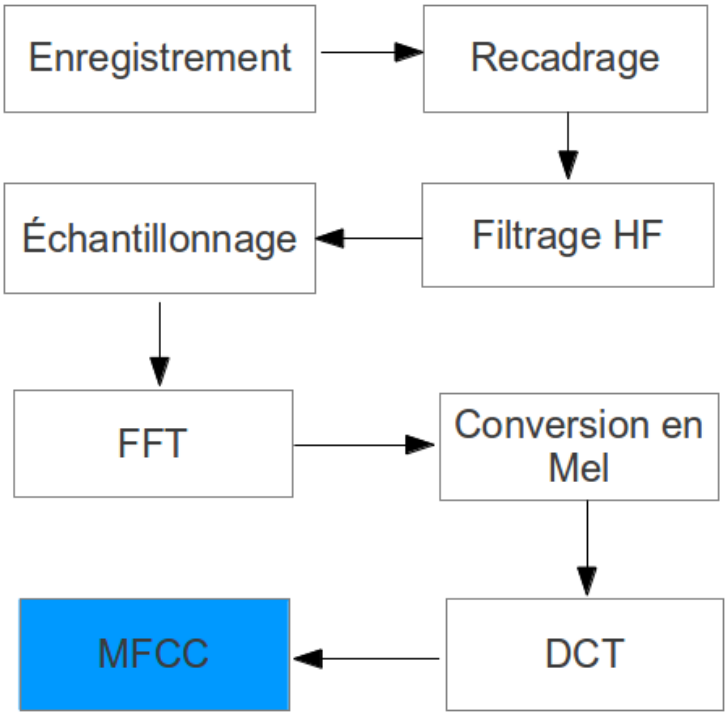

FIGURE 3.3 – exemple

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, élement 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

1.0.1 Choix d'une architecture optimale pour notre projet

Distribuer notre projet tel quel présenterait à ce stade de nombreux défauts : - le cœur 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 API) aves les serveurs de The SpeechApp Company. Ces derniers seuls auraient accès au cœur algorithmique du projet, qui resterait 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 hebergées serait monétisable. Alors, The SpeechApp Company pourrait prioritairement développer deux applications connectables au serveur : la première, SpeechCreator, 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 multitudes 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 :

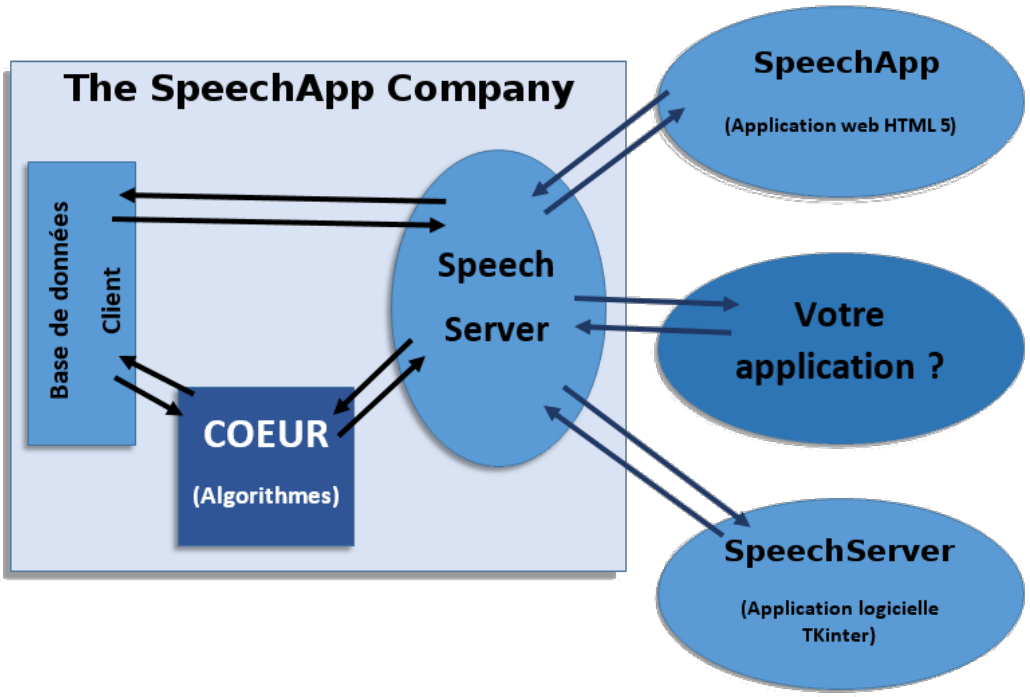

FIGURE 1.2 – Architecture proposée pour le projet The SpeechApp Company

Plus précisement dans le cadre des échanges entre le SpeechServer, les requêtes pourraient être traitées de la façon suivante : Le client (au sens logiciel toujours) envoie au SpeechServer une requête HTTP POST 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 cœur algorithmique, le SpeechServer répond au client par une réponse HTTP POST contenant des données au format XML. Le client peut alors lire et interpreter la réponse donnée par le SpeechServer.

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

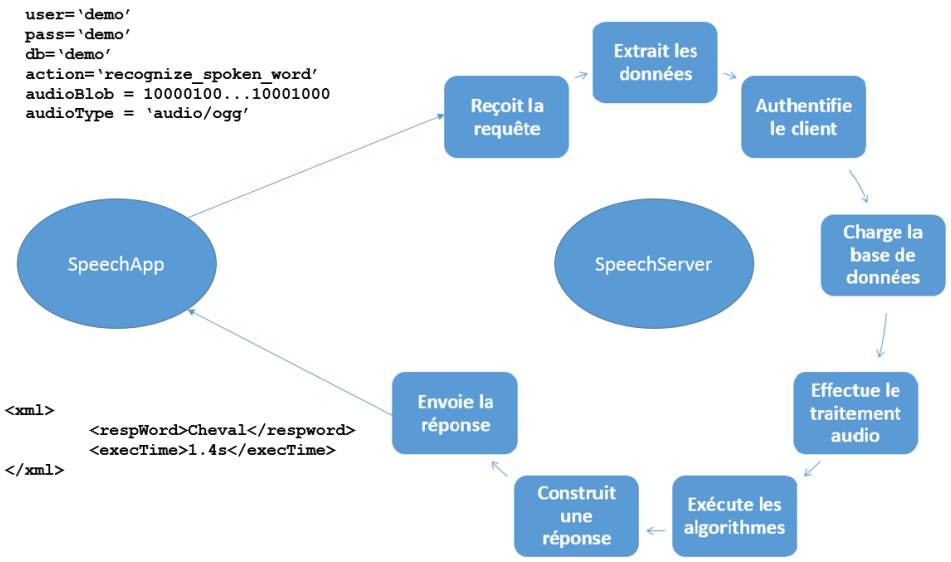

FIGURE 1.3 – Reconnaissance d'un mot par SpeechApp couplée au SpeechServer

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

— permettre la création d'un écosystème varié d'applications basées sur le cœur algorithmique de The SpeechApp Company via l'API de son SpeechServer, et générant ainsi des revenus
— conserver le cœur 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 cœur 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 a 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 cœur 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.

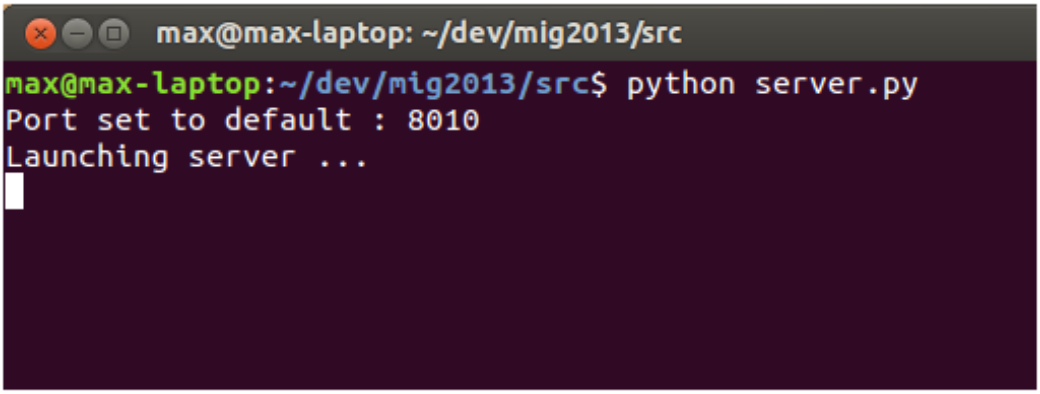

FIGURE 1.4 – Le SpeechServer lancé

Il écoute alors les requêtes sur le port 8010 (par défaut) de l'ordinateur. Lorsqu'il en reçoit, il interragit avec le cœur 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 leurs 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 developpé 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 dur, *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 Dimensionneemnt 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 $(10 * 100ko + 50ko) * N * M = 2.100$To d'espace sur le disque dur.

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

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 élements de calculs

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 simultannée 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 dédié au calcul atteigne ce niveau de performance. Le Intel Xeon E5-2670 atteint ainsi en théorie 330 Gflops

1.1.5 Choix de l'infrastructure et coûts liés

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 préférerons prendre le processeur bi-Intel Xeon ES-2670 Nous sélectionons 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 définis, 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 developpé 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

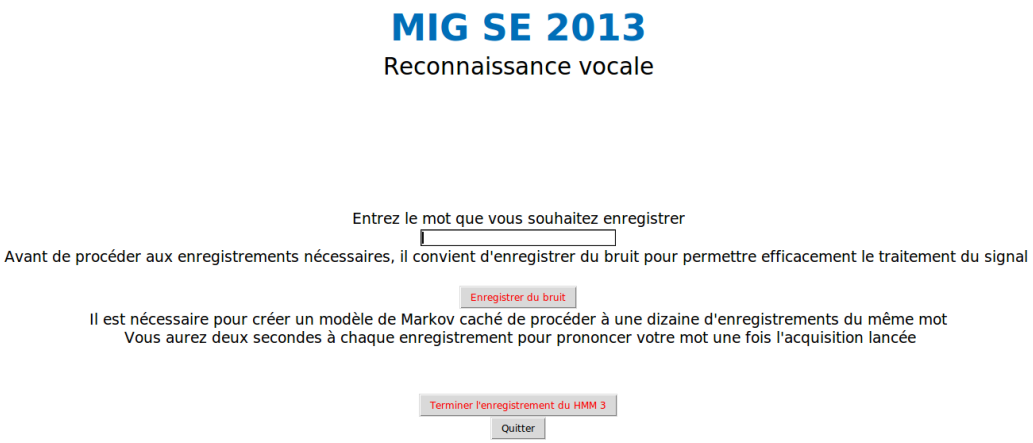

FIGURE 1.7 – L'interface du SpeechRecorder

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 conçu 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".

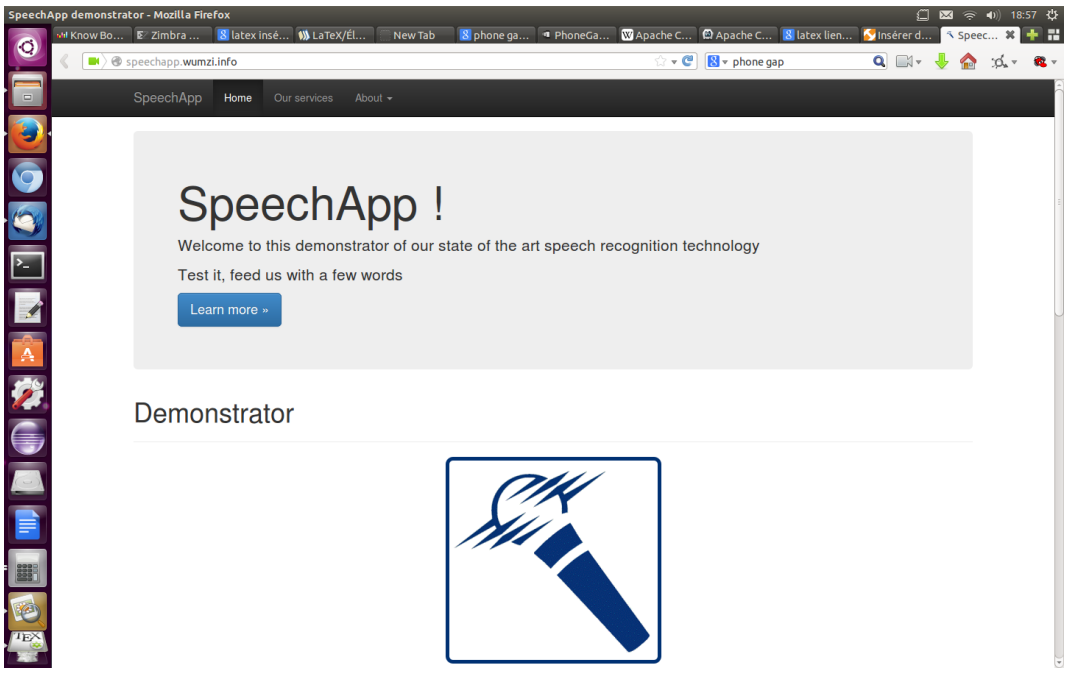

FIGURE 1.8 – Le démonstrateur SpeechApp sur ordinateur

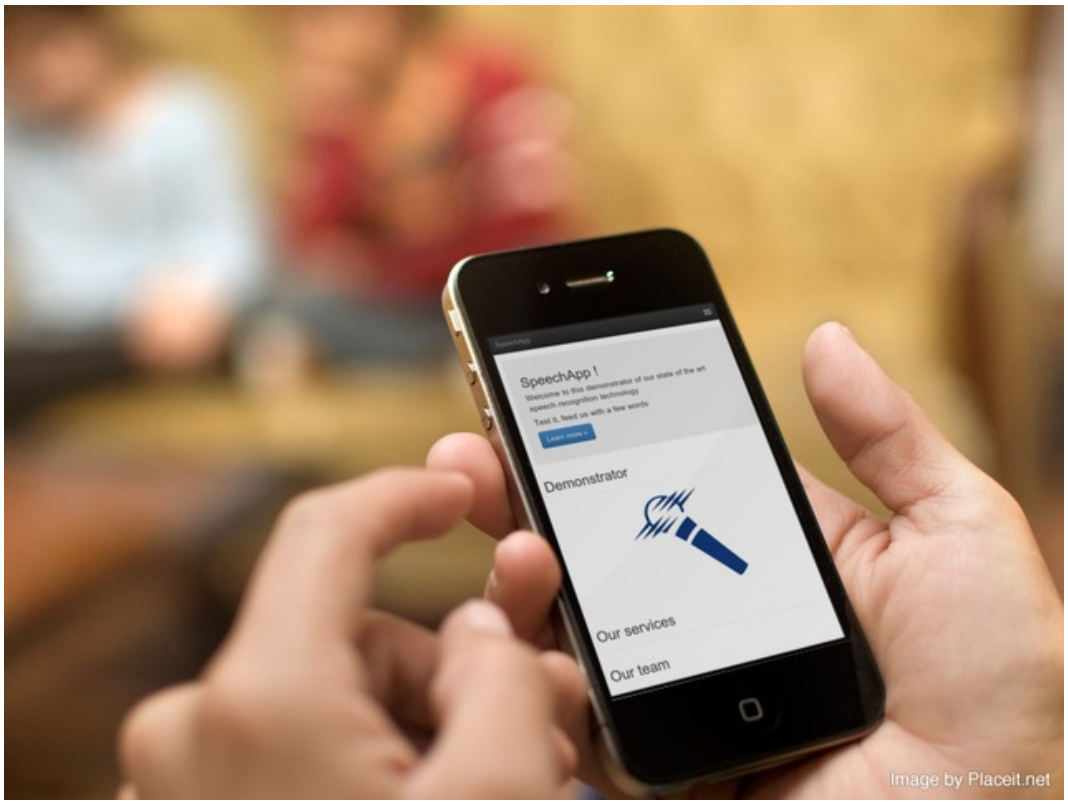

FIGURE 1.9 – Le démonstrateur SpeechApp sur IPhone

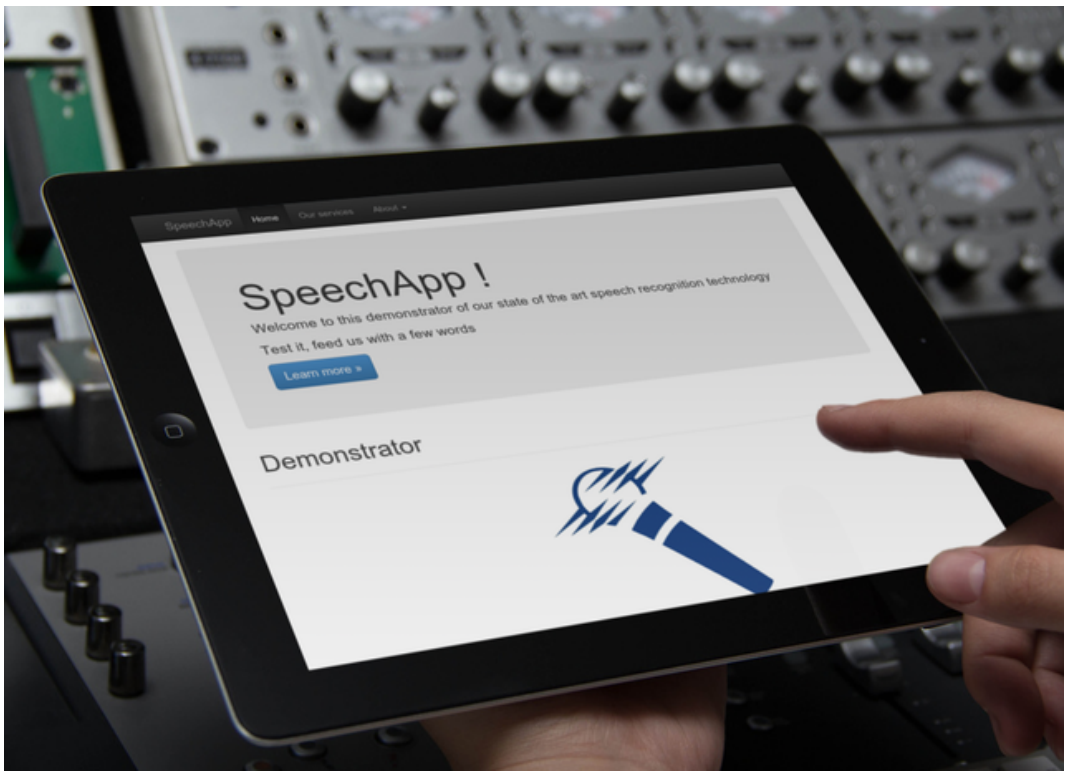

FIGURE 1.10 – Le démonstrateur SpeechApp sur IPad

Une difficulté néanmoins a été de rendre l'enregistrement audio fonctionnel sur la majorité des navigateurs. Nous assurons la compatibilité pour les moteurs Gecko et WebKit récents 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. Applications

La reconnaissance vocale est une technologie promise à un futur radieux ; les plus grands noms de l'informatique, dont Bill Gates, annonçaient il y a quelques années qu'elle allait remplacer les claviers d'ici peu. Il s'avère aujourd'hui que leurs prédictions ne sont pas encore réalisées, il est tout à fait possible qu'elle se réalise plus tard que prévu. Le principal obstacle à l'explosion de cette technologie étant le manque de fiabilité totale, mais avec les progrès à venir, la technologie deviendra de plus en plus sûre.

L'armée étatsunienne a bien compris le potentiel de cette technologie : elle investit massivement depuis des années dans la recherche pour la développer[13]. Elle est d'ailleurs déjà utilisée sur certains avions de chasse, et pas seulement aux Etats-Unis : en France, en Angleterre et en Suède aussi notamment. Vu les investissements massifs, il y a fort à penser que les armées de ces différents pays ont des techniques bien plus avancées que celles connues du grand public, qui sont déjà plutôt performantes. Pour le moment, les commandes vocales ne servent pas encore à des fonctions critiques comme lancer un missile, et elles 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. La technologie est également utilisée sur certains hélicoptères. Dans les deux cas, elle demande 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 elles demandent un savoir-faire qui est totalement hors de notre portée.

La reconnaissance vocale est également utilisée dans le contrôle aérien[14], 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, et elle 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.

La reconnaissance vocale se développe dans de nombreux domaines professionnels où les tâches administratives prennent beaucoup de temps, notamment la médecine, le droit et la police. En médecine[15], elle permet de remplir des rapports médicaux automatiquement : une simple relecture est alors nécessaire. Elle est notamment déjà utilisée dans 95% des hôpitaux aux Pays-Bas. Pour le droit, elle pourrait remplacer le travail du greffier pour prendre des notes dans les tribunaux. Et pour la police[16], 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 est souvent largement nécessaire. Dans le domaine du droit, il faut néanmoins prendre en compte les conditions particulières d'enregistrement (brouhaha ambiant, émotions dans la voix, volume variable...). Notre produit peut tout à fait servir à ce type d'applications, à condition de créer une base de données spécifique aux domaines concernés.

Une autre application possible de la reconnaissance vocale est l'aide aux handicapés[17], par exemple des commandes vocales pour une chaise roulante. Les phrases utilisées sont très typées (avancer, reculer,...) donc 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 peut rentrer les commandes ce qui assure un taux de reconnaissance très élevé. Notre produit peut donc bien s'adapter à cette utilisation.

La technologie est également très utilisée pour un usage plus ludique : fonctions de recherche dans les téléphones mobiles, les ordinateurs, robotique, jeux vidéo, traduction automatique,... Notre produit, dans sa version pour les particuliers, peut servir à ces usages même si la concurrence ne manque pas.

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 pour notre produit sont assez nombreuses, et la demande est de plus en plus forte, ce qui montre sa pertinence.

3. Budget, modèle économique

3.1 Introduction

Après les études techniques et théoriques, l'étude économique est une nécessité. Elle est au cœur 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.

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[18], 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 [19]. Sur ce salaire brut, l'employé paye environ 22% de charges salariales, et l'entreprise 44% de charges patronales.

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). 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'euros 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. 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 prémices 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 €[20].

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

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. 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

Le marché de la reconnaissance vocale est pour le moment assez restreint, mais est appelé à grandir dans les prochaines années. Si les systèmes de reconnaissance vocale fleurissent sur les objets multimédias à usage personnel, comme les ordinateurs portables ou les téléphones mobiles, ils servent uniquement à simplifier un peu certaines tâches de l'utilisateur, et ne sont en pratique que très peu utilisés, ce qui s'explique par leurs performances moyennes. Le représentant le plus utilisé de ce type d'usage de la reconnaissance vocale est probablement Siri sur les téléphones mobiles iPhone d'Apple, mais il reste assez peu utilisé malgré la grande popularité de l'iPhone.

Dans le domaine des logiciels payants, pour un usage plus sérieux, le marché est dominé par les logiciels Dragon NaturallySpeaking de la firme américaine Nuance. Les prix, selon les modèles, varient entre environ 100$ pour le modèle de base et environ 1000$ pour les versions spécialisées dans un domaine professionnel. Le principe est que plus la base de données de mots est grande, plus les erreurs sont fréquentes ; Dragon NaturallySpeaking[21] propose donc des versions adaptées à un domaine particulier. Par exemple, il existe une version "juriste" avec une base de données contenant surtout du vocabulaire technique de droit, et une version "médicine" avec des termes techniques médicaux. Ces versions visant une cible très précise donc plus restreinte, ils sont vendus considérablement plus cher que les versions plus classiques. 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 au plus est nécessaire. La réussite est renforcée par des taux de réussite exceptionnels avec des bases de données adaptées, et par l'absence de concurrence très forte.

Cependant, nous avons choisi de concevoir un logiciel avec une base de données moins spécialisée pour un usage personnel : en effet, dans le temps qui nous est imparti, créer des bases de données étudiées spécialement pour un certain domaine (droit, médecine) 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 eux-mêmes leur base de données spécifique à leur domaine. L'inconvénient de cette approche est le désagrément de devoir ajouter soi-même les mots, l'avantage étant que la reconnaissance sera plus fiable puisque la voix de l'utilisateur elle-même sert de comparateur, et elle permet d'avoir une base de données réellement personnalisée (celles de Dragon, bien que dédiées à un domaine, ne sont pas totalement personnelles). Le prix envisagé de la licence pour cette utilisation de notre produit est comparable (de l'ordre de 1000€) à celui des versions personnalisées de Dragon.

Nous prévoyons également de mettre en vente une version à usage personnel, sans possibilité d'ajouts de mots, au prix de 5€. Il est difficile de prévoir le potentiel de cette version, puisque les concurrents sont très nombreux, de qualité et de prix très variables.

Troisième partie

Core

A. Code Principal

A.1 shell.py

A.2 server.py

A.3 gui.py

B. handling

B.1 fenetre$_h$*ann.py*

B.2 inverseDCT.py

B.3 triangularFilterbank.py

B.4 passe$_h$*aut.py*

B.5 fft.cpp

C. HMM

C.1 creationVecteurHMM.py

C.2 markov.py

C.3 tableauEnergyPerFrame.py

C.4 hmm.cpp

D. recorder

D.1 recorder.py

D.2 sync.py

E. utils

E.1 animate.py

E.2 constantes.py

E.3 db.py

E.4 util.py

Quatrième partie

SpeechApp

.5 main

.6 holder

.7 recorder

.8 recorderWorker

.9 index.html

Cinquième partie

SpeechServer

.10 main.py

.11 audioConverter.py

.12 clientAuth.py

.13 speechActions.py

Bibliographie

[1] Apple. Application siri. http://www.apple.com/fr/ios/siri/, 2013.

[2] Tom Preston-Werner, Chris Wanstrath, and PJ Hyett. Github. http://www.github.com/, 2013.

[3] Lawrence Rabiner. *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1993.

[4] 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.

[5] Begam, Elamvazuthi, and Muda. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw). *Journal of Computing*, 2010.

[6] 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.

[7] 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.

[8] Chris Bagwell. Sox website. http://sox.sourceforge.net/Docs/Documentation, 2009.

[9] Anonyme. Théorème de nyquist-shannon. http://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_d%27%C3%A9chantillonnage_de_Nyquist-Shannon, 2013.

[10] Luc Maranget. *Introduction à la programmation*. École Polytechnique, 2008-2009.

[11] Maurice Charbit. Reconnaissance de mots isolés (utilisation des modèles HMM). *Inconnu*, Oct. 2002.

[12] Franck Bonnet, Benjamin Devèze, Mathieu Fouquin, and Julien Jeany. Reconnaissance automatique de la parole. *Epita*, 2004.

[13] Aharon Etengoff. Nuance clinches speech-recognition deal with us army. http://www.itexaminer.com/nuance-clinches-speech-recognition-deal-with-us-army.aspx, 2009.

[14] Thanassis Trikas. Automated speech recognition in air traffic control. *MIT*, 1987.

[15] G2 Speech. La reconnaissance vocale pour hôpitaux et autres institutions de soins. http://www.g2speech.be/reconnaissance-vocale.html, 2000.

[16] Nuance Company. Dragon naturallyspeaking for law enforcement. http://www.nuance.com/naturallyspeaking/industries/law-enforcement/, ?

[17] Nancy Manasse. Speech recognition. *University of Nebraska-Lincoln*, 1990.

[18] PrismaMedia. Capital. www.capital.fr, ?

[19] Michael Page. Hays, officeteam, 2009-2010.

[20] Gouvernement français. site des impôts. www.impots.gouv.fr, 2013.

[21] Nuance Company. Dragon naturallyspeaking. http://www.nuance.com/dragon/index.htm, ?