HEY MORTY!

EPITA

2ème année cycle préparatoire

Assistant vocal

Deuxième soutenance

Auteurs

BARONNET Matthieu CHASSIGNOL Valentin

CHEVREAU Annabelle GIRAUD Lise

16 juin 2021

Table des matières

Chapitre 1

Introduction

1.1 Le cadre

Comme chaque année à EPITA, les étudiants doivent réaliser un projet encadré. Cette année, Lise, Matthieu, Annabelle et Valentin se sont réunis pour réaliser un projet d'assistant vocal dont nous allons expliciter les spécifications ci-après. Ce projet se doit de répondre à un certain nombre de contraintes imposées comme la présence importante d'algorithmique.

1.2 Qu'est-ce qu'un assistant vocal ?

Un assistant vocal est un logiciel qui a pour but d'enregistrer une commande vocale provenant d'un utilisateur et d'exécuter la demande vocale. Ainsi, il est possible de décrire son fonctionnement en plusieurs étapes bien distinctes :

— Premièrement, la récupération de la commande vocale. C'est l'étape la plus importante. Afin que notre assistant vocal fonctionne correctement et puisse exécuter les ordres donnés, il doit d'abord bien les comprendre, la suite ne pouvant fonctionner correctement sans.

— Deuxièmement, l'interprétation d'une commande. Après avoir correctement enregistré ce qu'on lui demande de faire, l'assistant vocal doit "comprendre" ce que l'utilisateur désire, c'est-à-dire qu'il doit récupérer les mots-clés un par un (les segmenter) afin de prioriser chaque mot et savoir quoi faire.

— Dernièrement, l'exécution de la tâche demandée. Une fois tout le reste effectuée de façon correcte, il faut que notre assistant vocal soit en mesure de faire ce qu'on lui demande pour répondre à la requête de l'utilisateur.

1.3 Les contraintes

Les principales contraintes sont :

— L'ensemble du projet doit être écrit en C (C99), langage appris durant l'année

— Le projet doit être développé pour une plateforme Linux.

Chapitre 2

Le groupe BURP

2.1 Le nom & le logo

Notre groupe s'appelle BURP, le projet PICKLE. Notre assistant vocal se nomme Morty.

Et maintenant vous allez nous dire pourquoi ces noms ?

Le nom de notre groupe et de notre projet ainsi que celui de notre assistant vocal sont tous les trois inspirés de la série Rick & Morty. En effet, nous partageons une passion commune pour cette série et c'est donc pour cela que nous avons choisi ces noms. De plus, dans la série, Rick est un véritable génie inventeur et Morty est son acolyte, celui qui l'accompagne dans toutes ses aventures, c'est pour cela que nous avons nommé notre assistant Morty.

2.2 Les membres du groupe

2.2.1 Matthieu BARONNET

Hey ! Moi c'est Matthieu. J'ai commencé la programmation en 4ème en cours de technologie (il s'agissait de HTML et CSS) et depuis j'ai développé une certaine passion pour la programmation qui m'a amené à prendre l'option ISN en seconde et à apprendre le langage Python. Cependant je n'ai pas vraiment travaillé sur de vrai projet avant ma terminale en ISN où l'on nous a poussés à coder un Tetris ainsi qu'un système de chiffrement (bien que basique) de nouveau en Python.
Maintenant que je suis à Epita je peux assumer à 100% mon côté geek étant donné que tout le monde ici est autant ou plus geek que moi.

2.2.2 Valentin CHASSIGNOL

Je suis Valentin, aka. Vinetos. J'ai découvert la programmation en 2012 avec la découverte du jeu Minecraft. Cette rencontre avec la partie cachée du jeu permettant de modifier le comportement des serveurs multijoueurs et plus tard modifier le comportement du jeu en lui-même est devenue une véritable passion qui je l'espère deviendra une vocation.
Techniquement parlant, je suis plutôt très à l'aise dans la création et la contribution de projets informatiques. Je manipule plusieurs langages de programmation de manières régulières tout en participant à des projets open-source ce qui me permet d'avoir une expérience non-négligeable dans ce domaine.

2.2.3 Annabelle CHEVREAU

Hello ! Moi c'est Annabelle. Depuis ma découverte de l'informatique et plus précisément de la programmation en classe de première S-SI, je suis déterminée à accroître mes connaissances dans ce domaine. En classe de Terminale S-SI j'ai eu l'occasion de réaliser des petits projets, principalement en langage Arduino, tel qu'un petit robot qui roule en détectant les murs. J'ai de plus choisi l'option ISN qui m'a permis d'apprendre le C et de réaliser un jeu de petits chevaux en 2D, avec un langage appris de façon autodidacte, le processing. Déterminée, je participerai du mieux possible à la bonne réussite du projet.

2.2.4 Lise GIRAUD

Je suis Lise ! J'ai beaucoup de passions mais les choses qui m'occupent le plus sont le sport, la littérature et les jeux vidéo. Je me suis découvert une passion pour l'informatique et la programmation dès la troisième lors d'un défi entre les collèges de ma ville.
Cette passion nouvellement découverte, en seconde j'ai également pu participer à un projet en ISN où nous avons réalisé un site web et des petits robots en utilisant Arduino. Enfin, mon orientation en S-SI m'a permis de réaliser deux autres projets informatiques en première avec mon TPE (création d'un gant interactif permettant de supprimer l'utilisation de la souris) et ensuite en terminale, tous deux en C++.
Aujourd'hui je mêle toutes ces passions en étudiant à EPITA tout en me divertissant lorsque j'ai du temps libre.

Chapitre 3

Préambule du projet

3.1 Introduction du projet

3.1.1 Origine du projet

A l'origine nous avions de nombreuses idées de projet avant de nous décider sur l'assistant vocal. Parmi ces idées :

— Un moteur de recherche.

— Un langage de programmation (donc un compilateur pour un langage de notre invention).

Nous avons creusé la possibilité de travailler sur ces projets tout en cherchant d'autres idées qui nous intéresseraient plus ou qui colleraient mieux aux contraintes du projet de S4. Finalement, après une longue discussion l'idée de travailler sur de la reconnaissance vocale nous est venue.

Faire un assistant vocal n'était pas encore à l'ordre du jour, nous nous demandions s'il n'était pas trop complexe de faire un programme qui permettrait une reconnaissance vocale. Suite à de nombreuses blagues et références à des programmes tels que Cortana, Siri ou encore Jarvis (l'assistant de Tony Stark dans les films Iron Man), nous nous sommes trouvés forts intéressés par l'idée même de faire un assistant vocal. Elle reprenait la partie reconnaissance vocale qui nous intéressait et y ajoutait un moyen d'utiliser les résultats de celle-ci, rajoutant une profondeur nouvelle au projet qui nous a tous séduite.

3.2 Objet de l'étude

3.2.1 Les attentes et objectifs

La création de cet assistant vocal demande des connaissances dans divers domaines. De plus, travailler en équipe va nous demander une fois de plus de faire preuve d'organisation et de confiance envers les différents membres du groupe.
Nous allons devoir nous coordonner sur la façon d'écrire le code, donner des noms de variables explicites et surtout commenter au maximum nos fonctions afin qu'elles soient compréhensibles et modifiables par tous les autres membres en fonction des besoins, et par la suite pouvoir réunir les différentes parties aisément afin d'éviter de gérer d'éventuels problèmes inopinés.

3.2.2 Définition du projet

Nous souhaitons donc réaliser un assistant vocal. Il se découperait en plusieurs parties décrites ci-dessous :

— **Récupération de la commande vocale :** La première étape consiste à récupérer l'entrée micro de l'ordinateur afin de capter, lorsque que l'utilisateur le décide, sa commande vocale. Il faut donc enregistrer ce résultat à un endroit dans la mémoire afin de traiter son signal par la suite.

— **Transcription de la commande :** Une fois la commande vocale récupérée, il faut l'analyser en découpant chaque mot prononcé par l'utilisateur afin d'identifier clairement les mots-clés qui correspondent à des tâches à effectuer. Chaque mot sera ensuite transmis au réseau de neurones qui aura été préalablement entraîné à reconnaître des mots-clés.

— **Compréhension de la commande vocale :** Lorsque la commande vocale a été segmentée et identifiée, l'analyse des mots est alors nécessaire pour traiter la demande.
Il s'agit d'établir une priorité sur chaque mot de la phrase afin de comprendre quelle est la tâche à réaliser. Par exemple, dans une phrase, un mot tel que "exécute" a plus d'importance qu'une simple conjonction de coordination. Aussi, il est possible de reconnaître des formes de phrases prédéfinies (Ex : "ouvre ... ", "Lance ...", etc.).

— **Exécution de la tâche demandée :** Intervient enfin la dernière étape : réaliser l'action demandée par l'utilisateur. Il faut, par exemple, exécuter un programme qui ouvre la bonne application. Utilisant ainsi les fonctions vues lors du séminaire Unix pour lancer les processus nécessaires à l'exécution de la tâche souhaitée.

Afin de pouvoir développer chacune des parties du projet indépendamment les unes des autres, nous n'allons pas nous concentrer directement sur la réalisation du réseau de neurones pour reconnaître les mots prononcés. Il s'agit donc de créer et/ou d'utiliser dans un premier temps une librairie externe pour convertir de l'audio en texte et le traiter.

3.3 État de l'art

Le premier assistant vocal (ou virtual assistant) moderne installé sur smartphone fut Siri, qui a été introduit sur l'iPhone 4S le 4 octobre 2011 par Apple. Son but était d'aider à accomplir des tâches telles que l'envoi de message, faire des appels, vérifier la météo ou encore activer une alarme. Avec le temps il évolua pour donner des recommandations de restaurants, faire des recherches internet ou encore donner des directions en voiture.

En Novembre 2014, Amazon annonça la sortie de deux autres assistants virtuels : Alexa et Echo suivis par le Google Assistant de Google qui débuta en Mai 2016.

Les virtual assistants utilisent du traitement automatique des langues (natural language processing : NPL) pour lier une entrée texte ou une entrée audio (voix) pour exécuter des commandes. Beaucoup apprennent continuellement en utilisant des techniques d'intelligences artificielles, notamment le "Machine Learning".

Certains de ces assistants ont même des aptitudes supplémentaires telles que la reconnaissance d'objets à l'intérieur d'images (aussi appelée "images processing") afin d'aider l'utilisateur à obtenir de meilleurs résultats à partir d'image.

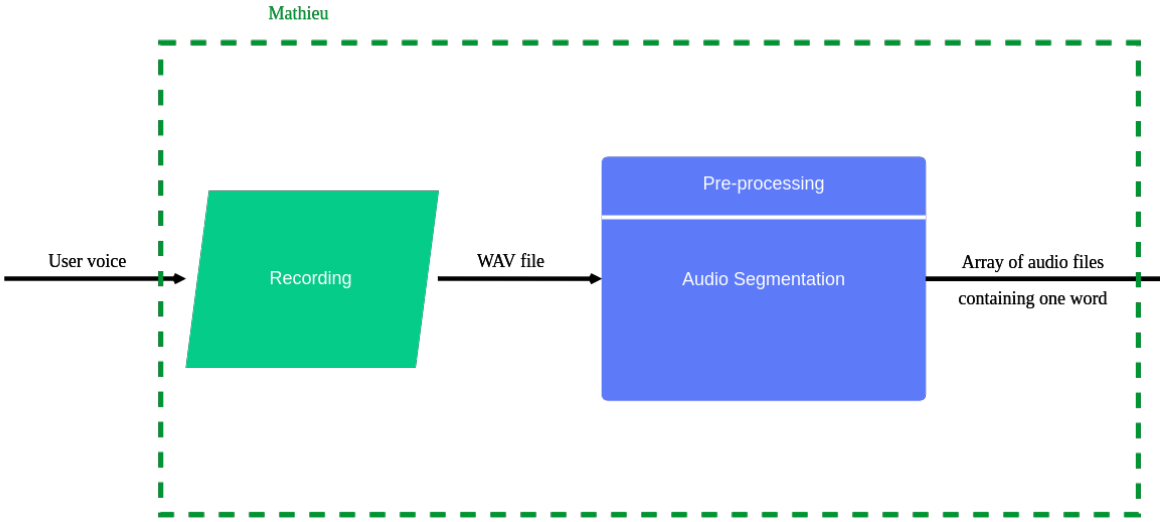
Dus à l'activation de ces assistants via des mots-clés et à la hausse en popularité de ces derniers, les risques, notamment légaux, deviennent des enjeux des plus importants à mesure que le temps passe.

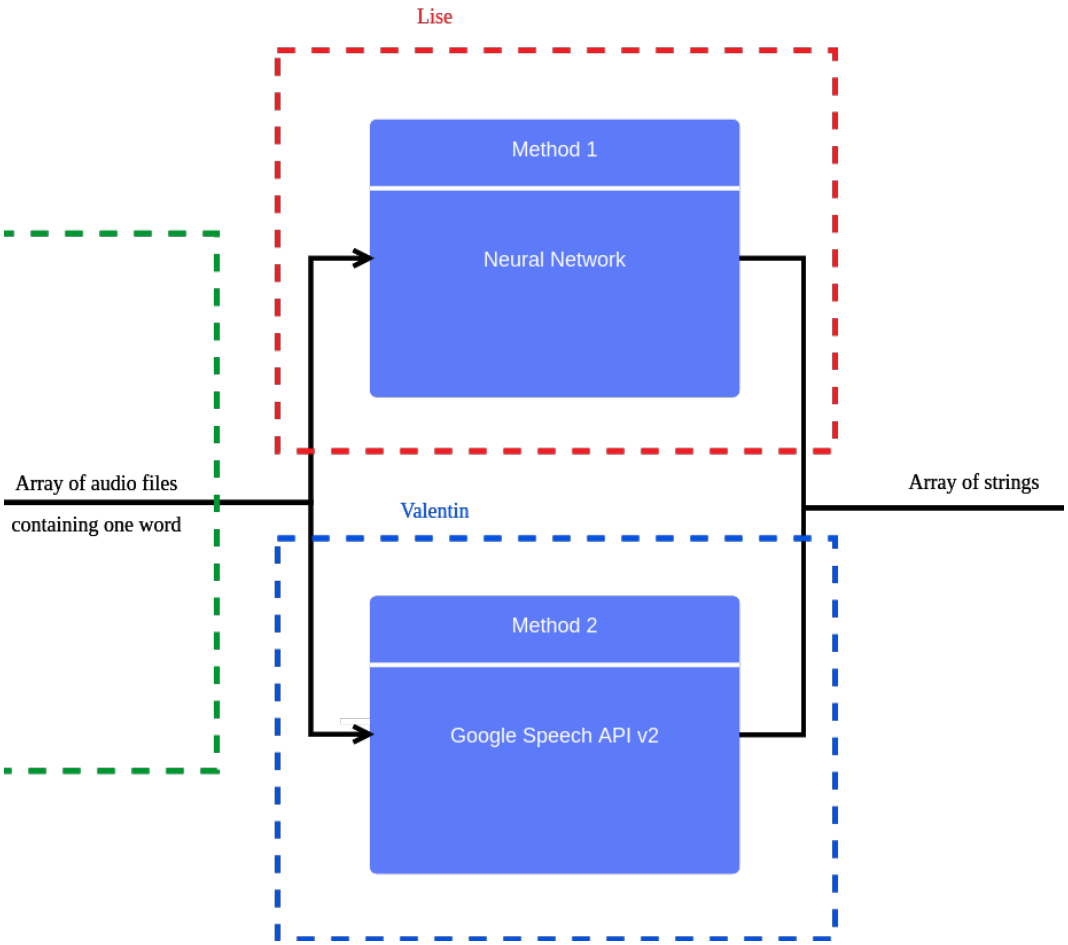
Chapitre 4

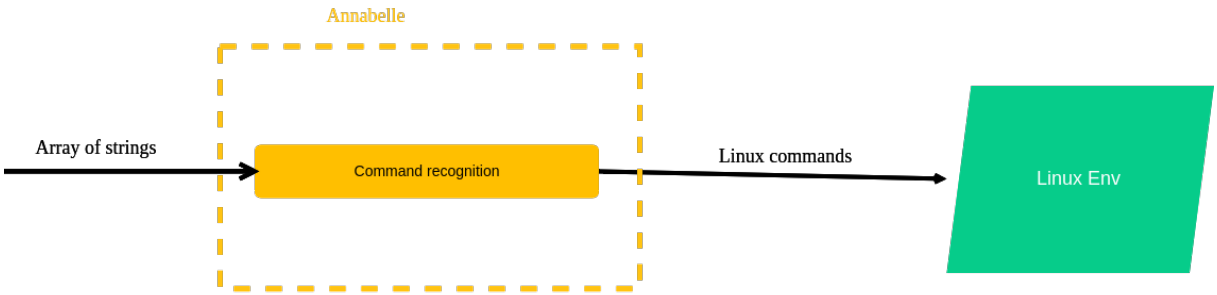
Présentation du projet

4.1 Définition / Trouver un nom

Un bon moyen de résumer l'organisation du projet peut se faire avec cette image.




Lise
Method 1
Neural Network
Array of audio files
containing one word
Valentin
Array of strings
Method 2
Google Speech API v2


Annabelle
Array of strings
Command recognition
Linux commands
Linux Env

4.2 Organisation du groupe

4.2.1 Github

Github, étant un pilier de l'open-source aujourd'hui, il fournit de bons moyens de gérer un projet. Il gère efficacement les branches, les pulls request, les merges, etc. C'est pour tout cela que nous avons décidé de travailler avec et ce, dans une dynamique de git flow. Ainsi nous allons utiliser une convention git commune pour nos commits. En effet, nous avons pris la décision de nous restreindre à la convention de git flow adaptée au niveau des commits.

Chaque commit se présente sous cette forme :

$$type : sujet$$

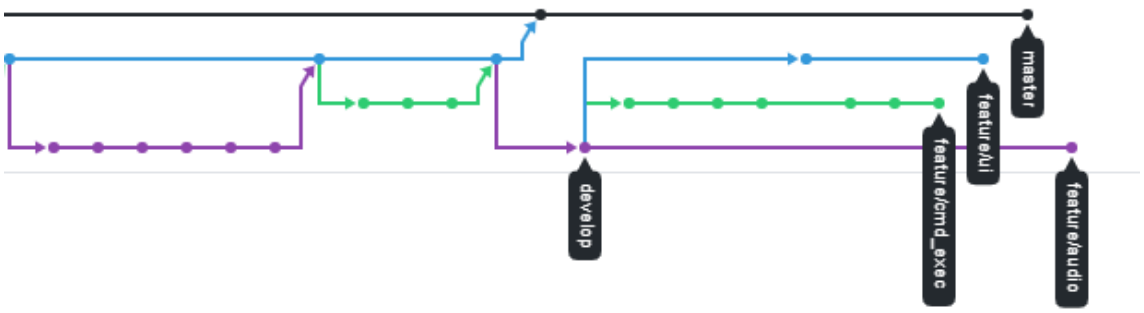
Type	**Description**
feat	Ajout d'une fonctionnalité
fix	Correction de bugs
docs	Ajout de documentation sur le projet (commentaire, Readme, ...)
refactor	Réécriture de code (amélioration)

Cela nous permet d'avoir un historique bien plus propre. Aussi, nous avons fait le choix de travailler sur des branches par fonctionnalité respectant ce format :

$$type/sujet$$

Branche	**Description**
master	Branche de production
develop	Next release
feature/<name>	Ajout d'une fonctionnalité
fix/<name>	Correction de bugs

Ce qui nous donne un réseau comme ceci :



4.2.2 Répartition des tâches

	Matthieu	**Lise**	**Annabelle**	**Valentin**
Capture / Transcription de l'audio	Resp.	Assist.		
Réseau de neurones		Resp	Assist.	Resp
Compréhension / Traitement de la commande	Assist.		Resp	
Site Web		Resp		Assist.

4.2.3 Communication dans l'équipe

Afin de communiquer entre nous sur les tâches à faire nous avons mis en place différents moyens de communication. Nous avons un serveur Discord pour les informations importantes, un groupe Messenger pour discuter de tout et de rien. Nous avons aussi une organisation GitHub pour partager les différents projets, fichiers et les documents et voir les tâches de chacun. De plus, nous nous voyons de façon quotidienne ce qui nous permet d'échanger directement sur nos avancées et/ou les problèmes rencontrés. [1]

4.3 Programme soutenances

Voici le planning des soutenances que nous avons établi. Toutefois, il est possible que ce planning soit sujet à des modifications ultérieures en fonction de l'avancement du projet.

	Soutenance 1	Soutenance 2	Soutenance Finale
Capture/Trait. de l'audio	25%	75%	100%
Réseau de neurones	30%	75%	100%
Trait./Exécution des commandes	25%	75%	100%
Site web	20%	80%	100%

4.4 Avancement à la première soutenance

Pour rappel, à la première soutenance nous avions un léger retard global sur notre projet. Cependant, nous avions décidé de ne pas modifier notre planning d'avancement puisque nous pensions que notre retard serait rattrapable. Toutefois, à la deuxième soutenance, nous avons toujours du retard de façon générale mais nous ne modifions toujours pas notre planning d'avancement puisque malgré tout nous espérons mener a terme notre projet d'ici à la dernière soutenance.

1. https ://github.com/Pickle-Burp

Chapitre 5

Capture et traitement de l'audio

5.1 L'objectif

Le but de la capture et du traitement de l'audio est de récupérer le son, et donc la phrase de l'utilisateur, via une entrée audio (par exemple un microphone), de l'enregistrer en tant que fichier pour le transformer en spectrogrammes et enfin l'envoyer au réseau de neurones.
En réalité la tâche est bien plus compliquée que cela :
il faut d'abord ouvrir une entrée pour le son, c'est-à-dire déterminer le nom de l'appareil utilisé pour la capture et le stream (celui de capture). Ensuite on détermine les paramètres du format d'enregistrement afin d'obtenir un fichier WAV.
Une fois le fichier obtenue nous devons récupérer le tableau de nombre correspondant à l'amplitude du son enregistré. Nous pourrions directement donner ces nombres au réseau de neurone mais essayer de reconnaître des mots à partir de ces échantillons directement s'avère être une tache très difficile. De ce fait nous devons faire un travail de pré-traitement sur nos données audio pour rendre le problème plus facile pour le réseau de neurone. L'étape de pré-traitement nécessitera de produire à partir du fichier audio un spectogramme.

5.2 Fichier WAV

Vu que l'on enregistre d'abord en format WAV, nous devons suivre certaines conditions d'enregistrement :

- il faut utiliser un PCM entrelacé non compressé (uncompressed interleaved PCM),
- les échantillons seront de taille 16 bits,
- en utilisant un petit endian (little endian),
- sur 2 canaux,
- avec des échantillons pris sur des intervalles de 16 000 Hz

5.3 Traitement de l'audio

Maintenant que nous avons le fichier audio il nous faut le transformer pour pouvoir le rendre utilisable. À cet étape nous avons un fichier binaire encodé sur 16 bits qui n'est pas exploitable. Pour pouvoir traiter cet audio il faut créer à partir de ce binaire un tableau de nombres représentant l'amplitude à chaque 1/16'000 de seconde du son enregistré. Cependant si l'on venait à observer graphiquement une représentation de cette amplitude, elle montrera la combinaison complexe de plusieurs fréquences de son qui forment la parole. Désormais il nous faut transformer ce tableau en un spectogramme utilisable par le réseau de neurones. Cette partie nécessitera l'application d'une fonction "fenêtre d'observation" ainsi qu'une transformation de Fourier sur le résultat de la fonction précédente. Généralement les fenêtres sont sur une durée de 20ms pour permettre la détection de lettre, cependant vu que le réseau de neurone traite des mots nous pouvons augmenter la taille de cette fenêtre par plus de 10, augmentant la précision de notre spectogramme sur l'axe des fréquences. Ainsi une fenêtre optimale serait d'une taille de 300-400ms. Ceci n'est clairement pas la méthode idéale pour traiter des mots et non pas des lettres et nécessitera sans doute plus de recherches.

Chapitre 6

Analyse 1 - Réseau de neurones

6.1 Qu'est-ce que c'est ?

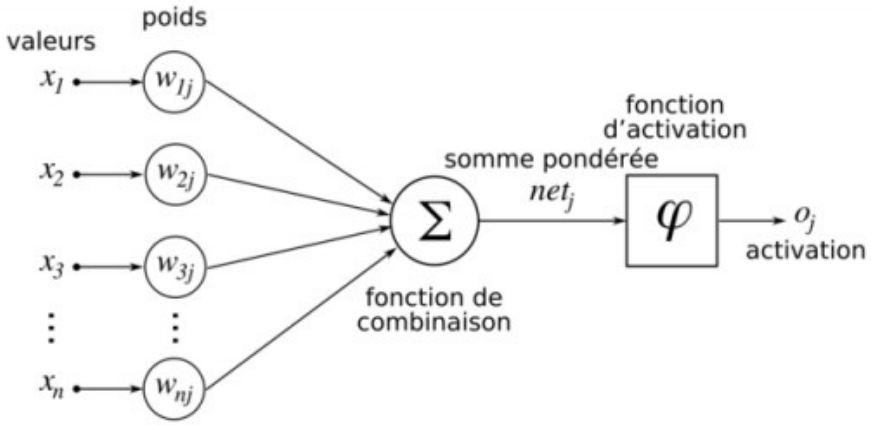
Le réseau de neurones est un élément de l'assistant vocal très important car, en effet, c'est lui qui sera capable d'identifier / classifier les mots afin de reconstituer la commande voulue. Cette partie est compliquée à mettre en place car le réseau est une approximation du fonctionnement du cerveau : il est capable d'apprendre pour se perfectionner et ainsi avoir des résultats de plus en plus près de la vérité.
En fait, à l'inverse des algorithmes classiques qui définissent un traitement sur la donnée en entrée, on fournit le résultat attendu et on laisse l'algorithme faire le lien entre l'entrée et la sortie attendue.

Les réseaux de neurones sont constitués de plusieurs couches (au minimum deux) :

- couche d'entrée : les neurones de cette couche contiendront l'information que l'on souhaite tester
- couche de sortie : ces neurones nous donneront le résultat du problème que le réseau aura résolu
- une ou plusieurs couche(s) cachée(s) : dans cette couche les neurones serviront seulement pour le calcul, leurs valeurs ne seront exploitées que par le réseau de neurones.

Les neurones de la couche d'entrée sont chacun reliés à tous les neurones de la première couche cachée (s'il y en a une) ; ceux de la première couche cachée sont reliés à tous ceux de la couche suivante, et ainsi de suite jusqu'à celle de sortie. Il peut arriver que les neurones d'entrées soient connectés à la couche de sortie bien qu'il y ait des couches cachées entre, ce qui permet d'améliorer l'apprentissage.

Chaque connexion entre deux neurones possède un poids qui permet de faire les différents calculs. Chaque neurone possède aussi une valeur de sortie (comprise entre 0 et 1) qui est la valeur à tester pour la couche d'entrée et le résultat d'un calcul pour les autres couches, cette valeur correspond à l'information que chaque neurone transmet aux neurones suivants.



6.2 Implémentation du réseau de neurones

Une fois la structure choisie et mise en place nous avons pu nous consacrer aux caractéristiques du réseau de neurone.
N'étant pas familiers avec les réseaux de neurones, il a été décidé que pour la première soutenance un réseau de neurones "d'entraînement" serait implémenté. Ainsi, ce dernier est capable de reconnaître toutes les portes logiques que l'utilisateur lui donne (AND, XOR, OR, NAND, ...).
Afin de bien comprendre le fonctionnement d'un réseau de neurones, nous avons fait en sorte que l'utilisateur puisse choisir lui-même le nombre de layers, le nombre de neurones pour chaque layers, le taux d'apprentissage (learning rate) et le set d'entrainement (qui représente u ne porte logique de son choix).

6.3 Propagation

La propagation désigne le mécanisme qui permet au réseau de propager l'information au travers de ses couches successives, de la couche d'entrée à la couche de sortie, et sans avoir un retour de l'information en arrière, alors il s'agit d'un réseau à propagation avant ou "feed-forward" en anglais. Les exemples les plus connus sont le perceptron simple et sa version multi-couches.

La première étape de la propagation est de donner les valeurs d'entrées à notre réseau de neurones. Pour cela on déclare que les valeurs des neurones de la couche d'entrée sont égales à nos valeurs d'entrées du réseau.

$$a_{\ell,1} = g\left(\sum_i w_{\ell-1,i,1}\, a_{\ell-1,i}\right)$$

FIGURE 6.1 – Formule de calcul d'une valeur de neurone

Par la suite, pour propager les valeurs dans la couche suivante il faut pour chaque neurone : appeler une fonction d'activation qui prend en entrée le biais du neurone actuel additionné à la somme de l'ensemble des multiplications des valeurs des poids qui relie ce neurone aux neurones de la couche précédente par les valeurs des neurones de la couche précédente. Dans notre cas, nous avons décidé d'en utiliser 2 principales pour notre réseau d'entraînement (ces deux fonctions sont aussi les plus couramment utilisées) :

6.3.1 Fonction Sigmoïde

Pour la dernière couche, nous avons choisi d'utiliser la fonction sigmoïde qui renvoie une valeur située entre 0 et 1 sur le principe de la formule :

$$S(x) = \frac{1}{1 + e^{-x}}$$

FIGURE 6.2 – Fonction sigmoïde

Et une fois ce calcul effectué, le neurone prend la nouvelle valeur qui lui est affectée.

6.3.2 Fonction Unité Linéaire Rectifiée

La fonction Unité Linéaire Rectifiée (Rectified Linear Unit ou ReLU en anglais) est appliquée à toutes les couches intermédiaires du réseau de neurones (Hidden layers). Cette fonction renvoie 0 si l'entrée est négative et sinon elle renvoie la valeur donnée en entrée. Le principal avantage de cette fonction est sa simplicité de calcul, en particulier pour sa dérivée. Il y a également moins de problème de gradient par rapport à la fonction sigmoïde qui a tendance à saturer dans les deux sens.

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$$

FIGURE 6.3 – Fonction ReLU

Ensuite, comme pour la fonction sigmoïde, une fois le calcul effectué, le neurone prend la nouvelle valeur donnée par la fonction.

6.4 Rétro-propagation

En statistiques, la rétro-propagation du gradient est une méthode pour calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers

la première. De façon abusive, on appelle souvent technique de rétro-propagation du gradient l'algorithme classique de correction des erreurs basé sur le calcul du gradient grâce à la rétro-propagation et c'est cette méthode qui est présentée ici. En vérité, la correction des erreurs peut se faire selon d'autres méthodes, en particulier via le calcul de la dérivée seconde.

La technique du gradient descendant consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs. Dans le cas des réseaux de neurones, les poids synaptiques qui contribuent à engendrer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale.

Ce principe fonde les méthodes de type algorithme du gradient, qui sont efficacement utilisées dans des réseaux de neurones multi-couches comme les perceptrons multi-couches. L'algorithme du gradient a pour but de converger de manière itérative vers une configuration optimisée des poids synaptiques. Cet état peut être un minimum local de la fonction à optimiser et idéalement, un minimum global de cette fonction (dite fonction de coût).

Normalement, la fonction de coût est non-linéaire au regard des poids synaptiques. Elle dispose également d'une borne inférieure et moyennant quelques précautions lors de l'apprentissage, les procédures d'optimisation finissent par aboutir à une configuration stable au sein du réseau de neurones.

6.4.1 Algorithme

Les poids dans le réseau de neurones sont au préalable initialisés avec des valeurs aléatoires. On considère ensuite un ensemble de données qui vont servir à l'apprentissage. Chaque échantillon possède ses valeurs cibles qui sont celles que le réseau de neurones doit à terme prédire lorsqu'on lui présente le même échantillon. L'algorithme se présente comme ceci :

— Soit un échantillon x que l'on met à l'entrée du réseau de neurones et la sortie recherchée pour cet échantillon t.
— On propage le signal en avant dans les couches du réseau de neurones. On étudie le taux d'activation des neurones de la dernière couche.
— Ensuite en connaissant le taux d'activation que ces neurones devraient avoir, et via cela, on modifie les valeurs transportées par les neurones et leurs liens.
— Répété un grand nombre de fois, ce processus permet au réseau de neurone de s'améliorer et de réaliser la tâche qui lui incombe.

6.5 L'apprentissage

L'apprentissage est le cycle complet de propagation/rétro-propagation répété un certain nombre de fois. Ce nombre est appelé époque (ou "epoch" en anglais).

6.6 Recherches liées au réseau de neurones

Cependant, avant d'implémenter quelque réseau de neurones que ce soit, il nous fallait en savoir plus sur ce dont nous aurions besoin pour notre projet.
En effet, après quelques recherches rapides nous nous sommes vite aperçus qu'il existait en réalité plusieurs types de réseaux de neurones différents, chacun étant plus ou moins efficace pour certains types de traitement.
Ainsi, nous avons donc fini par trouver que les catégories suivantes de réseaux de neurones étaient les plus efficaces pour notre assistant vocal : les CNN (Convolutional Neural Network), les DNN (Deep Neural Network) et les RNN (Recurrent Neural Network). D'après la page anglaise de Wikipédia sur le deep learning, les taux d'erreur peuvent varier de façon significative en fonction du réseau de neurones utilisé :

Method	Percent phone error rate (PER) (%)
Randomly Initialized RNN[130]	26.1
Bayesian Triphone GMM-HMM	25.6
Hidden Trajectory (Generative) Model	24.8
Monophone Randomly Initialized DNN	23.4
Monophone DBN-DNN	22.4
Triphone GMM-HMM with BMMI Training	21.7
Monophone DBN-DNN on fbank	20.7
Convolutional DNN[131]	20.0
Convolutional DNN w. Heterogeneous Pooling	18.7
Ensemble DNN/CNN/RNN[132]	18.3
Bidirectional LSTM	17.8
Hierarchical Convolutional Deep Maxout Network[133]	16.5

FIGURE 6.4 – Résumé des taux d'erreur mesurés depuis 1991 sur 630 anglophones américains pour des réseaux de neurones différents (base de donnée TIMIT)

L'image ci-dessous, nous indique donc l'efficacité des différents types de réseau de neurones sur le même set d'entraînement pour de la reconnaissance vocale. Il est important de noter que le réseau de neurones avec le taux d'erreur le plus faible (Hierarchical Convolutional Deep Maxout Network ou HCDMN) est en réalité un CNN qui utilise la fonction d'activation maxout au lieu de l'habituelle fonction sigmoïde ou ReLU utilisées sur ce genre de réseau de neurones en plus d'avoir une structure légèrement différente des CNN habituels (sa structure est constituée d'une hiérarchie, celle du modèle de Hidden Markov, Hidden Markov modèle en anglais).

Enfin, d'après cet article [1], les HCDMN semblent être plus puissants que les DNN dans certains cas et sont plus "souples" quant à l'entrée donnée. En effet, ils peuvent tolérer des légères fluctuations en entrée et toujours être capable de donner la bonne sortie. Il a été montré que le gain de performance fourni par ce modèle est dû à sa structure particulière et pas simplement à cause du contexte d'entrée plus large et de l'utilisation de plus de couches entre l'entrée et la sortie. De plus, la fonction maxout peut également être facilement combinée avec la fonction de mise en commun des neurones convolutifs. Ainsi, les CNN avec une fonction maxout et un modèle d'Hidden Markov sont plus efficaces que les CNN classiques (avec la fonction sigmoïde ou ReLU). Il est donc suggéré que les RNN et CNN sont les deux types de réseaux de neurones qui peuvent permettre d'obtenir les meilleurs résultats pour la reconnaissance vocale.

6.6.1 Les réseaux de neurones à convolution

Après plusieurs recherches, nous avons décidé que nous utiliserions un Convolutional neural network (CNN) pour notre assistant vocal. En effet, il s'avère qu'il sera plus simple de traiter notre signal sonore en tant qu'image et les CNN sont majoritairement utilisés pour le traitement d'images, ce qui les rend assez performants lorsqu'on les utilise pour du son. Ainsi, une fois que le son est transformé en image (spectrogramme) il est envoyé dans le CNN et ce dernier commence alors son analyse.

Architecture

Les CNN sont composés d'une architecture qui peut s'avérer assez complexe mais qui les rend d'autant plus performants dans le traitement de signal.

Couche de convolution Il est important de noter que dans un CNN, l'entrée est un tenseur. Un tenseur est outil mathématique qui "généralise les matrices". Par exemple, cela permet d'ajouter une couche en "profondeur" sur une matrice 2D. Dans notre cas, il s'agit d'un tenseur composé de :

(nombre d'images) x (hauteur des images) x (largeur des images) x (canaux des images).

La couche de convolution traite les données d'un champ récepteur, mais surtout c'est le bloc de construction de base d'un CNN. Le volume de la couche de convolution dépend des trois paramètres suivants :

- **La profondeur** : C'est le nombre de neurones associés à un même champ récepteur.
- **Le pas :** Il contrôle le chevauchement des champs récepteurs. Ainsi, plus il est petit et plus le chevauchement des champs récepteurs sera élevé et le plus le volume de sortie sera grand.
- **La marge :** Elle permet de contrôler la dimension spatiale du volume de sortie. La formule pour calculer le nombre de neurones du volume de sortie est celle-ci :

$$W_0 = \frac{W_i - K + 2P}{S} + 1$$

  Avec W_i la taille du volume d'entrée, K le nombre de champs récepteurs, S le pas et P la taille de la marge.

1. https ://asmp-eurasipjournals.springeropen.com/articles/10.1186/s13636-015-0068-3#Sec12

L'image ci-dessous schématise une couche de convolution avec un ensemble de neurones (cercles) qui créent la profondeur d'une couche de convolution (en bleu). Ils sont liés au même champ récepteur (en rouge).

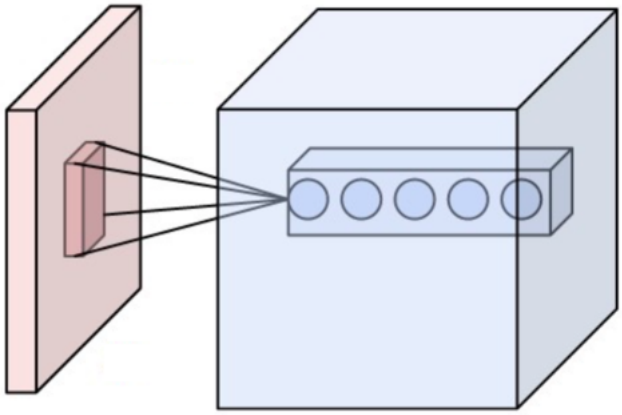


FIGURE 6.5 – Couche de convolution

Couche de pooling Le pooling est le moyen de mettre en commun et résumer les informations extraites de l'image par la couche de convolution. En effet, cette dernière, dans le meilleur des cas, conserve la taille des données en entrée et dans le cas général l'augmente considérablement. La couche de pooling fait alors un sous-échantillonage de l'image le plus fidèle possible dans le but de simplifier considérablement le nombre d'élément tout en gardant au mieux les particularités.

Pour cela, on découpe les différentes images renvoyées par la couche de convolution en une série de carrés de $n \times n$ pixels qui ne se chevauchent pas. Pour chaque carré, on calcule la valeur de pooling. Cette valeur représentera et remplacera notre carré de n^2 valeurs.

Il existe plusieurs fonctions de pooling, toutes avec leurs avantages et inconvénients. Les plus utilisées sont :

— **Max-pooling :** elle prend la valeur maximum du carré dans lequel on se trouve et renvoie cette valeur.

— **Average-pooling :** elle calcule la moyenne des valeurs de tous les pixels présents dans le carré actuel et renvoie cette moyenne.

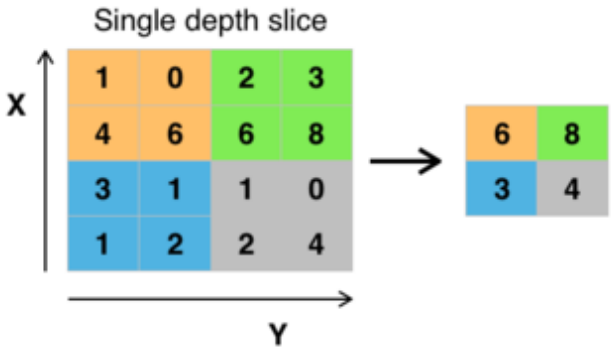


FIGURE 6.6 – Schéma symbolisant un "max-pooling" avec un filtre de taille 2×2.

Il est important de souligner que la couche de pooling n'est pas obligatoire dans un CNN mais elle permet de gros gains en puissance calcul puisqu'elle réduit la taille de la représentation de l'image : un carré de taille n^2 devient 1 valeur. Cependant, cette réduction entraîne bien évidemment une perte d'informations et il convient donc d'utiliser des petits filtres (de 2x2 ou 3x3 suffisent) même si cela augmente le nombre de calculs.

Il est bien sûr possible de supprimer la couche de pooling mais cela pourrait causer un sur-apprentissage de notre réseau de neurones et une augmentation des calculs à effectuer.

Couche de correction Afin d'améliorer l'efficacité du traitement on intercale une couche de correction entre les couches de traitement. Cette couche est en réalité une fonction d'activation (cf partie 6.3).

Couche entièrement connectée Après plusieurs couches de convolution et de max-pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente (comme on le voit régulièrement dans les réseaux réguliers de neurones). Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

Couche de perte La couche de perte spécifie comment l'entraînement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. En voilà une liste non exhaustive :

— La fonction Softmax

— La descente de gradient

— Erreur quadratique moyenne

Ces différentes couches se suivent et constituent le modèle d'apprentissage traditionnel des CNN. Il faut noter qu'il est possible de répéter les couches de convolution, max pooling et correction autant de fois qu'on le souhaite. Plus ces couches sont répétées plus le traitement du réseau de neurones sera précis mais cela demandera plus de calculs et plus de temps. Il existe bien sûr d'autres modèles d'apprentissage (comme celui cité dans la partie 6.6) qui influent sur l'efficacité du réseau de neurones.

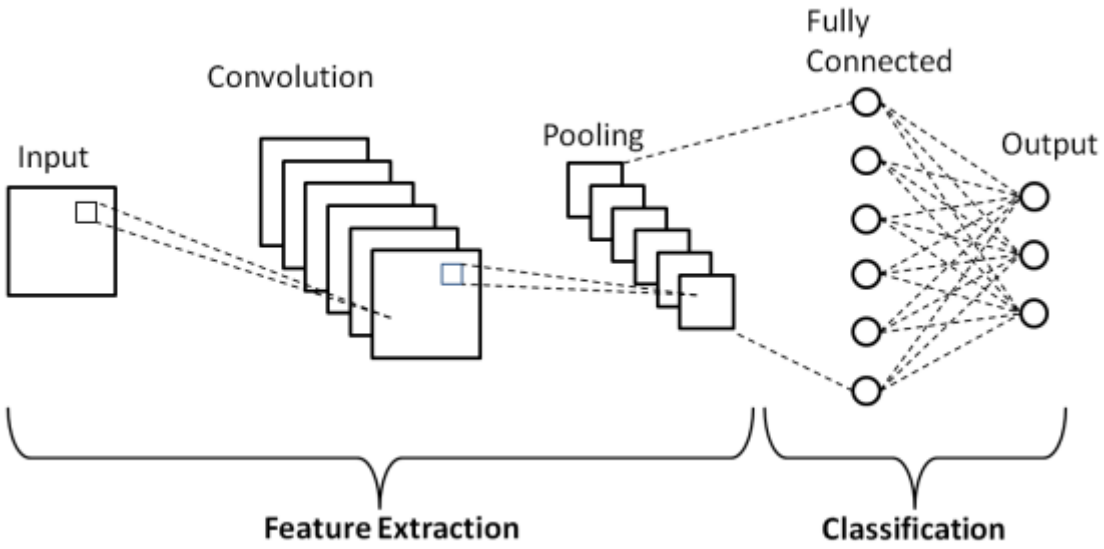


FIGURE 6.7 – Schéma récapitulatif pour le fonctionnement d'un CNN avec un modèle d'apprentissage traditionnel.

Avantages

La raison pour laquelle nous avons choisi d'utiliser un CNN plutôt qu'un autre type de réseau de neurones c'est avant tout parce qu'il comporte plusieurs avantages non négligeables. En effet, le principe même de convolution permet d'effectuer le pré-traitement nécessaire à la bonne analyse du son par le réseau de neurones. Cela implique qu'il n'y a que très peu de paramétrage initial à faire et très peu de supervision à réaliser au cours du processus d'apprentissage et d'analyse. C'est donc un atout majeur des CNN.
De plus, un des avantages majeurs des CNN est l'utilisation d'un poids unique sur tous les neurones d'un même noyau de convolution, contrairement aux perceptron multicouche qui lui, peut affecter plusieurs poids sur un même neurone, ce qui augmente considérablement la mémoire nécessaire au fonctionnement du réseau de neurones.

La base d'entraînement

Un point très important lors du développement d'une intelligence artificielle est sa base d'entraînement. Sur ce point, il existe de nombreuses bases de données de mots et phrases en français faites par différentes organisations principalement universitaires.

Mais, un projet tout neuf se démarque du lot : Mozilla Common Voice. Très récent, il s'agit d'un projet collaboratif de création d'audio et de validation de ces derniers. Son principal avantage est qu'il regroupe 682 heures d'enregistrements dont 623 validées avec un peu moins de 13 000 voix différentes. De plus, le projet fournit sur son site internet une interface plus que simple d'utilisation pour enregistrer ou valider des samples.

Le seul point négatif est que se projet est très peu organisé. Il contient simplement un fichier TSV qui labelise les fichiers audios qui ne sont que de simple numéro.

Chapitre 7

Analyse 2 - APIs Google

Notre projet ayant comme constituant un réseau de neurones (défini ci-dessus) pour transformer la capture audio de la commande vocale en chaînes de caractères, il nous faut le temps de son développement (ou en cas d'échec, en remplacement du réseau) un moyen de pouvoir faire cette conversion. Après plusieurs recherches, il existe assez peu de librairies ou API qui permettent de réaliser ce que l'on souhaite réaliser.

Après un certain nombre d'essais peu fructueux, nous avons choisi d'utiliser Google Cloud Speech API qui se veut d'être particulièrement complète. Il s'agit de cette même API qu'utilise YouTube. Son utilisation est plutôt simpliste : on envoie l'audio encodé dans une requête JSON et le serveur nous renvoie ce qu'il pense être le texte prononcé ainsi que plusieurs alternatives avec leurs taux de confiance respectifs.

Il a fallu créer un client HTTP avec libcurl qui se charge de construire et faire la requête puis en récupérer le résultat. Mais, parce que tout n'est pas aussi simple, il était nécessaire d'avoir des identifiants pour obtenir des résultats. Ayant comme contrainte de ne rien devoir installer sur les PCs, il est donc impossible d'utiliser cette même API qui requiert d'installer le SDK google cloud.

Heureusement, en fouillant bien, il existe une vieille API qui n'est pas mentionnée officiellement qui correspond à nos besoins : pouvoir faire une requête juste avec un identifiant. Les seuls points négatifs est qu'elle n'est pas aussi performante que celle officielle et qu'elle ne supporte que des formats audio très précis : FLAC en stéréo en 44kHz ou WAV Mono en 16kHZ.

Cette plateforme nécessitant d'être authentifié et d'avoir installé le Google Cloud SDK, il existe une version non-publiée de l'API qui est accessible avec un simple compte développeur Google. Une requête contenant l'audio, la langue et la clé suffit pour obtenir un résultat, certes moins précis, mais suffisant.

Chapitre 8

Analyse 3 - Deepspeech

Mozilla DeepSpeech est un moteur Speech-To-Text open-source, utilisant un modèle formé par des techniques d'apprentissage automatique basées sur le document de recherche Deep Speech de Baidu. Le Project DeepSpeech utilise TensorFlow de Google pour faciliter la mise en œuvre.

Sa particularité réside dans sa puissance et dans sa portabilité. En effet, l'une de ses applications est de pouvoir être mise dans des systèmes embarqués HORS-LIGNE et de ne consommer qu'un minimum de ressource tout en donnant un résultat très précis.

Dans notre cas, il s'agit de la dernière alternative que nous pourrons utiliser dans le cas où l'assistant est utilisé dans un environnement hors-connexion.

Chapitre 9

Traitement/exécution des commandes

9.1 Traitement de la commande

Le traitement de la chaîne de caractères à la sortie du réseau de neurones est une étape essentielle dans la finalisation du projet. En effet cette étape consiste à transformer une chaîne de caractères, celle donnée par l'utilisateur, en commande, compréhensible par la machine.
Afin de simplifier la partie, du moins à ce stade du projet, il a été convenu que seules quelques commandes seraient possibles et sous une certaine forme uniquement.
Les entrées possibles sont donc comme ceci :
Les étapes entre parenthèses sont facultatives.

- pour ouvrir une application : "ouvre" + (l'application) + le nom de l'application
- pour lancer une recherche, plusieurs formulations peuvent être utilisées :
  - "recherche" + intitulé de la recherche. Qui utilise le navigateur par défaut de l'utilisateur pour mener à bien la recherche.
  - "recherche sur/dans" + nom du navigateur + intitulé de la recherche.
  - "recherche" + intitulé de la recherche + "sur/dans" + nom du navigateur. Ces deux dernières formulations sont similaires. Elles utilisent le navigateur précisé par l'utilisateur pour mener à bien la recherche.
- pour voir le contenu d'un dossier : "Montre-moi le contenu du dossier" + nom du dossier
- pour voir le contenu d'un fichier : "Montre-moi le contenu du fichier" + nom du fichier

D'autres entrées pourront être définies plus tard, par exemple, on peut imaginer que Morty puisse nous donner la météo, mettre un minuteur...
Un grand nombre de techniques peu fructueuses ont été testées. D'autres sont simplement à l'état de recherches car pour la plupart elles sont très difficiles à mettre en oeuvre, par exemple, un réseau de neurones capable de reconnaître et associer un type de phrase à une commande ou encore une analyse statistique des différents mots de la phrase prononcée par l'utilisateur.

L'implémentation actuelle prend seulement en compte les cas cités au dessus.

Le code fonctionne de la manière suivante :
Première étape : Chercher le mot clé dans la phrase qui est ensuite comparé aux mots clés suivants : "ouvre", ("montre-moi",) "recherche". Ce sont ces mots qui définissent la suite des opérations.
Pour chacun de ces mots, le traitement est différent.
Pour le mot "ouvre", on vérifie si le mot "application" est présent dans la phrase, s'il l'est on lance la commande correspondant au mot juste derrière le mot "application" sinon le mot juste derrière "ouvre.
Pour le mot "recherche", on vérifie si on trouve les mots, "sur"/"dans", dans la phrase. Si aucun de ces mots n'est présent, on se contente de trouver le navigateur par défaut pour ensuite pouvoir réaliser la recherche sur ce dernier. Si un de ces mots a été trouvé, on regarde où il se situe dans le tableau pour pouvoir concaténer le mot suivant à notre commande qui est par conséquent le nom du navigateur, auquel on concatène "google.com/search ?q=" ainsi que l'intitulé de la recherche entre guillemets. Pour trouver l'intitulé de la recherche il suffit de prendre la fin du tableau, si le mot "sur"/"dans" se situe juste après "recherche" sinon on prend tous les mots se situant entre le mot "recherche" et le mot "sur"/"dans". Il est nécessaire de concaténer "google.com/search ?q=" avant le début de l'intitulé de la recherche car sans le navigateur essaye de trouver un site internet lié au(x) mot(s) reconnu(s). Ce qui nous donnerai "maison.com", soit un site éventuellement a l'oppose de la recherche originale, alors que l'utilisateur recherche des liens en relation avec une maison voire une image de maison...

9.2 Exécution de la commande

L'exécution de la commande a été une partie plutôt simple.
En effet, une fois que le traitement de la commande est réalisé, nous avons une chaîne de caractères. Cette chaîne de caractères est ensuite passée à une fonction : *system*.
Elle s'utilise de la manière suivante :
*system(const char *command)* et renvoie un entier.
Une première vérification nécessaire est *system(NULL)*, si cette fonction renvoie 0, aucun terminal n'a été détecté, aucun terminal n'est donc utilisable pour exécuter la commande donnée.
Sinon, on peut lancer la fonction *system* avec pour paramètre la chaîne caractère traduite à l'étape précédente. Si elle renvoie un entier différent de 0, la commande n'a pas pu être exécutée.

Chapitre 10

Interface utilisateur

Pour cette deuxième soutenance, l'interface utilisateur est très simple.
Elle est composée de 4 blocs :

- Un petit texte de présentation.
- Une partie pour renvoyer les résultats d'une éventuelle recherche par l'utilisateur.
- Un bouton "Aide", comme son nom l'indique, une fois enclenché ce bouton permettra à l'utilisateur de comprendre comment utiliser notre assistant vocal. Il contiendra bien évidemment les prototypes des phrases possibles, les actions que notre assistant vocal est en capacité de réaliser.
- Un bouton, contenant un micro, qui une fois enclenché, lance le programme de reconnaissance et traite la demande formulée par l'utilisateur.

Chapitre 11

Site web

11.1 Environnement et outils

Le site web n'étant pas une priorité pour la deuxième soutenance, il ne contient que des briques de bases.

Note site web utilise de toutes nouvelles technologies pour son développement :

- Node.JS pour la gestion de dépendance, l'intégration du serveur Web permettant le rechargement à chaud (= en direct) des modifications
- Vue.JS, un Framework Javascript open-source et léger conçu pour construire des interfaces utilisateur et des applications web monopage
- Bulma, un Framework CSS morderne et open-source basé sur Flexbox qui a été un vrai coup de cœur.[1]. Sa grosse particularité est qu'il fonctionne sans aucun script JS. Il peut donc s'intégrer de partout.
- Une batterie d'autre extensions comme webpack pour développer des sites avec Node.JS, eslint pour le formatage, Github Action pour son déploiement automatique, etc.

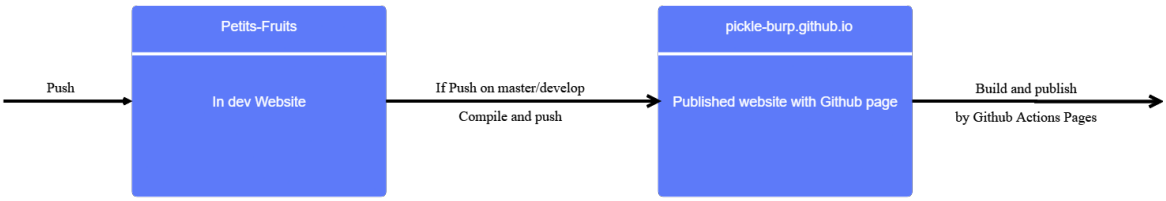
L'environnement de développement du site est assez complet et par extension assez lourd, pour cela le site utilisé est séparé en deux parties :

- la partie de développement du site (contenant des outils pour les tests, la vérification du code, un serveur web avec rechargement à chaud...) qui est contenu dans le projet « Website » sur Github.
- Une partie compilée, prête pour la production qui génère les fichiers HTML, CSS et JS correspondants.

11.2 Déploiement automatique

Nous avons mis en place un système un peu particulier pour le déploiement : À chaque push sur le repository « Website », le site est « compilé » et est exporté pour la production dans le repository pickle-burp.github.io afin d'être publié automatiquement par Github Pages.

1. Le dérivé de Bulma pour Vue.JS est Buefy, il est maintenu par la communauté



Après un push, un premier script est lancé pour compiler et tester le site. Puis, si tout va bien, il renvoie les modifications sur le projet auto-géré (pickle-burp.github.io) qui va ensuite se charger de déployer les modifications et des les rendre disponibles au grand public.

Pour cette première soutenance, le site ne contient rien. Simplement quelques composants et boutons afin de vérifier qu'il se déploie bien et que tout fonctionne.

11.3 Limitations

Un problème assez important lors du deploiement du site sur Github Pages est que si l'utilisateur clique sur un lien interne (par exemple /download), la page pourra s'afficher sans problème. Mais, si l'utilisateur demande la ressource sans passer par la page principale alors il va avoir une erreur 404. Pour corriger ce problème, il faudrait que le site soit configurer dernière un serveur web comme Apache2 ou Nginx.

Dans le cadre de notre projet, ce n'est pas très important car il s'agit d'un site vitrine. Ce problème ne sera donc pas résolu.

Chapitre 12

Conclusions

12.1 Ce qu'il reste à faire, retard/avance

De façon globale, notre projet a pris du retard. En effet, bien que nous sachions qu'il était ambitieux, nous ne nous attendions pas à autant de difficultés sur nos différentes parties. Ainsi, nous sommes en retard sur quasiment toutes les parties, mais ce n'est pas un gros retard. Ainsi nous espérons pouvoir rattraper ce retard pour la prochaine soutenance dans un mois environ. Voici la liste des tâches commencées :

— La capture et le traitement de l'audio ont complètement changé d'optique, bien que les entrées soient les mêmes, cette partie traite aussi la segmentation audio et la formation de spectogramme ce qui est loin d'être simple et rapide. Il reste beaucoup à faire avec ou sans aide de bibliothèque externe.

— Le réseau de neurones définitif n'est pas implémenté, seul un réseau de neurones "d'entraînement" existe. Cependant, les recherches nécessaires avaient été commencées à la première soutenance mais ont été approfondies et nous sommes aujourd'hui fixés sur ce que nous devons faire pour implémenter notre réseau de neurones.

— Le traitement et l'exécution des commandes ont été bien avancées mais doivent être encore être terminés.

12.2 Ressenti global

Pour conclure, l'avancement général du projet progresse lentement mais sûrement. Nous sommes toujours en retard et espérons que nous pourrons terminer ce projet dans les temps. Cependant, le travail de recherches est terminé ou presque, il ne reste donc plus qu'à implémenter nos algorithmes.
Toutefois, cela n'a pas impacté notre motivation et notre retard nous motive d'autant plus à travailler sur le projet pour le terminer dans les temps.