Rapport sur le stage effectué dans le projet Jeanne du 01/04/2021 au 03/06/2021

21809928 Antoine Chauveau
21706499 Tristan Dauty
21803128 Nicolas Poncet
21810195 Nelson Lefebvre

Lien dépôt git sur la forge :
https://forge.info.unicaen.fr/projects/jeanne-stage-crealog-sujetsubstitution?jump=welcome

Remerciements

Tout d'abord nous aimerions remercier toutes les personnes qui nous ont épaulé durant ce projet. Nous sommes partis sans aucune idée de ce à quoi nous allions devoir nous confronter. C'est donc dans la plus parfaite ignorance du sujet que nous nous sommes lancés. Clarifions quelques points avant de continuer, nous avons réalisé le projet par nos propres moyens (et avons été les seuls à effectuer des choix et ce en parfaite autonomie). Ceci étant si nous avons eu le luxe de choisir dans quelles voies aller c'est bien parce que des personnes nous ont renseignés et aiguillés dans des domaines d'expertise qui ne sont pas les nôtres mais aussi dirigés ou redirigés vers des personnes qui avaient des réponses, des connaissances à même de nous faire avancer.

Les premiers remerciements vont donc à Mr Lambert qui en plus de nous encadrer tout au long du stage a été à notre écoute et a fait au mieux pour nous rediriger vers des spécialistes.

- Mr Lambert Jean-Luc
- Mr Lefevre Franck
- Mr Mercouroff Nicolas
- Mme Lambert Natacha
- Mr Magarotto Eric
- Mr Routure Jean-Marc
- Mr Cagniot Emmanuel
- Mr Fougeray Pascal
- Mr Spaniol Marc
- Mr Cook Jason

Table des matières:

5- Analyse syntaxique.

a) Recherches
b) Pistes explorées
c) Choix et implémentation

6- Deep learning, réseau de neurones.

a) Définition du Deep learning
b) Introduction aux réseau de neurones
c) Exemple d'une librairie

7- Optimisation ajoutée (parallélisme).

a) Problèmes & Résolutions
b) Gestion des fichiers

8- Etat du produit au rendu de stage.

9- Sources.

1) Historique du projet

Mise en contexte :

Dans le cadre du projet de créativité logiciel nous devions imaginer un concept de produit. Nelson a pris l'initiative de proposer un assistant vocal qui fonctionnerait sans internet.

En effet, dans notre contexte où certaines personnes peuvent se retrouver isolées, nous cherchions une alternative de proposer un produit intuitif qui fournirait des connaissances et des informations. Car les personnes isolées peuvent se sentir délaissées de plusieurs façons. Elles peuvent ne pas avoir accès à internet ou aussi sentir un fossé entre elles et la technologie. Notre but était donc de réconcilier ces personnes avec "les nouvelles technologies".Tristan a alors suggéré de baser le concept sur les ondes radio.

Pourquoi la radio ? Simplement parce qu'elle est largement sous-estimée par la population. En général, elle est vue comme une technologie dépassée. Avec toutes les innovations technologiques tel que l'antenne 5G. On délaisse des technologie alors très élaboré, dans une logique de "sur-performance" en dépit de leur pertinence. Et plutôt que d'utiliser tous leurs potentiel on préfère s'en détourner pour aller vers les plus attrayantes, sans se poser la question de la pertinence effective de cette technologie.

Les ondes radio ont des particularités pourtant très intéressantes, les premières sont surtout la couverture qu'elles offrent ainsi qu'une infrastructure déjà présente dans une grande majorité du globe ainsi qu'une fiabilité éprouvée. Elles sont facilement accessibles et ce gratuitement. Mais surtout la radio est un médium de communication très puissant, véhiculant des informations diverses mais surtout pratique et le fait quotidiennement. Nous pouvons grâce à cela récupérer des informations tel que la météo, l'heure ou le trafic sans accès à internet et cela à n'importe quel moment. Et à l'opposé des informations certes nombreuses, un peu trop, présentes sur internet la fiabilité des informations à la radio sont bien plus grandes. Sans oublier que cette technologie est familière d'un grand nombre de personnes, tout particulièrement les personnes âgées.

Notre équipe est composé d'anciens membres de la matière de créativité logiciel :

- Antoine Chauveau
- Nicolas Poncet
- Tristan Dauty
- Nelson Lefebvre

Les Principales Phases du développement

1ère Phase	● Création du Trello (Base d'organisation du travail en phase). ● Contacter les professeurs et professionnels pouvant nous aider. ● Création du Google doc recensant toutes les informations récupérées. ● Choix du mauvais module (erreur entre transmetteur et récepteur). ● Création de la base de l'assistant vocal (écouter des phrases orales/restituer des phrases à l'oral). ● Optionnel : ajout de Wikipedia.
2ème Phase	● Commande du bon module (récepteur tea5767) ● Mise au point des ajout et modification avec Mr.Lambert ● Début du montage du prototype ● Ajout de lecture de musique en interne
3ème Phase	● Rendez vous en visioconférence avec Mr Lefevre Franck , un des quatre associé qui dirige la start-up développant Barnabé un assistant vocal pour sénior ● Analyse d'un jingle pour démarrer l'enregistrement ? ● Début de recherche sur le deep-learning et ce qu'il nous faudrait pour notre projet.

	• Recherches sommaire sur l'analyse syntaxique (Mail à monsieur Spaniol) : NLP (Natural Language Processing.
4ème Phase	• Rendez-vous avec Mercouroff Nicolas de la start-up tivine (contact donné par Mr Lefevre) qui analyse et utilise des flux venant de la télévision pour le client. Ce qui permet aux utilisateurs de récupérer des informations comme le produit présenté dans l'émission etc.. • Approfondissement du travail sur le deep learning suite à notre discussion avec Monsieur Mercouroff. (contact avec Mme Lambert Natacha pour plus de détails) • Développement plus poussé de l'analyse du sujet d'un flux audio hors deep learning. • Ajout de fonctionnalités assistant vocal.
5ème Phase	• Débug code, optimisation (parallélisation). • Rapport de stage. • Recherches sur le RDS. • Rencontre avec Jason Cook pour régler notre problème vis à vis de la radio FM • Approfondissement des maigres connaissances en électronique avec Mr Cook et découverte du ESP32 et MUSE PROTO (ESP32) qui pourrait être un meilleur support pour le projet. cf: raspiaudio.com

2) Raspberry et Hardware

a) Recherches sur la radio et ses composants

Notre premier objectif a été de comprendre comment fonctionnait une radio et de quoi elle était composée en termes d'électronique. Nous avons donc découvert les mécanismes qui nous permettent d'écouter la radio notamment avec un traitement de signal analogique capté par notre antenne qui va se voir convertie en données numériques grâce à un module électronique qui s'appelle un ADC. Ensuite ces données numériques peuvent être manipulées/analysées par plusieurs modules différents tel qu'un DSP (digital signal processor) ou un FPGA (field programmable gate away). A noter que le DSP est essentiellement utilisé pour le traitement du signal alors que le FPGA est un composants programmable qui peut accueillir un DSP (rend le produit assez cher mais il devient extrêmement performant) et qui de plus permet le parallélisme. Ensuite on renvoie ces donnés numériques dans un DAC (convertisseur numérique vers analogique) pour pouvoir sortir le flux audio à travers un haut parleur ou enceinte.

b)Recherches de projets radios

L'étape d'après à été de trouver comment monter une radio pour pouvoir l'utiliser et pouvoir y ajouter le code nécessaire à notre projet. Nous avons découvert énormément de projets de radio réalisés avec un Arduino. Nous avons regardé beaucoup d'entre eux et noté les modèles utilisés et les modules ajoutés pour avoir une radio fonctionnelle. Nous avons trouvé ensuite d'autres projets mais cette fois ci fait avec Raspberry. En parallèle de ces recherches nous avons envoyé un mail à plusieurs professeurs de l'université pour savoir s' ils pouvaient nous en dire un peu plus sur nos découvertes et nous conseiller. Nous avons contacté Mr Magarotto assez féru en électronique , ainsi qu'a MR Cagniot et MR Fougeray.

Mr Magarotto nous a redirigé vers des collègues professeurs d'électronique par manque de temps. Mr Routure nous à donc un peu aiguillé et conseillé sur les modules que nous lui avons montrés.

Nous avons donc émis deux hypothèses pour le matériel:

- Raspberry (Quelques pin pour réaliser de l'électronique, languages : Python , Beaucoup de librairies)
- Arduino (Très bon pour de l'électronique, Langage : C, mais n'a pas de processeur assez performant pour réaliser des calculs)

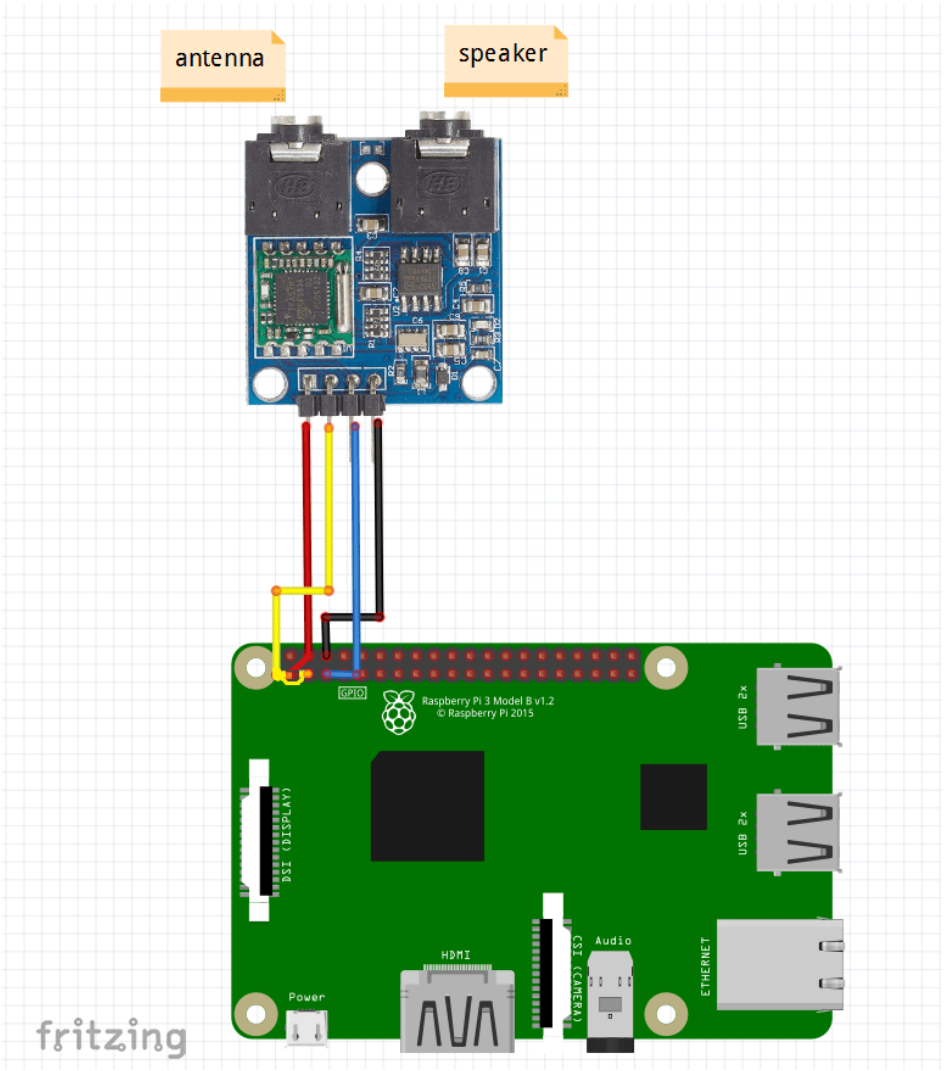
c) Choix matériel

Notre but était donc de réaliser un produit avant tout. Il fallait donc trouver un moyen de capter la radio tout en exécutant un code permettant de réaliser les tâches d'un assistant vocal et d'utiliser différents algorithmes d'analyse audio pour reconnaître dès que l'on met une onde radio sur écoute . (Ex: extraire l'audio parlant de météo)

Nous nous sommes donc tournés vers un raspberry car la seule contrainte que celui-ci avait était le peu de pin et un petit manque de présence de la communauté pour la partie électronique. Contrebalancé par une bibliothèque extrêmement fournie et variée à même de pouvoir nous fournir un gain de temps non négligeable dans la conception d'un modèle fonctionnel.

Pour la partie radio fm nous avons réalisé plusieurs recherches. Et nous nous somme tournés vers le module TEA5767 avec le tuto de Piddlerintheroot.
https://www.piddlerintheroot.com/fm-radio-tea5767/

Montage :



Néanmoins la correspondance entre nos programmes python sur raspberry et sur pc est compliquée. En effet, le raspberry possède un processeur fonctionnant sous ARM64. Or nos processeurs (sur la majorité des pc actuels) sont en x86-32 ou x86-34. Ce qui rend parfois incompatible certaines technologie ou librairie python.
De plus, le raspberry a besoin normalement de fonctionner sur du 15v et non du 5v. La majorité des chargeur/alimentation micro usb sont en 5v. Nous pensions avoir acheter un 15v mais celui-ci au final ne l'est pas.. Et en acheter un sur le web nous prendrait trop de temps à arriver. Dans l'état actuel le raspberry n'est donc pas capable de faire tourner l'assistant vocal. Et nous devons donc dissocier l'assistant sous un pc classique et la radio qui fonctionne sur raspberry .

Mais dans notre conception de produit final nous envisageons un produit comprenant :

- Un contrôleur (Raspberry)
- Un micro (Micro par usb)
- Un Module radio (tea5767)
- (Optionnel comme pour l'amazon echo show) Un écran pour potentiellement ajouter une interface graphique

- (SAA6588 pour capter le rds car après consultation de la doc, le module radio doit avoir ce module en plus.)

Après discussion au fab-lab avec un prof d'électronique/beau art, “Jason cook” nous avons réussi à trouver notre problème de bip constant. Notre branchement était mal réalisé et il ne s’agissait pas d’une question de voltage (on ne peut alimenter le raspberry en 15v que par . Et celui-ci nous a expliqué les différences entre différents microcontrôleurs et pc disponibles pour réaliser notre projet. Mais aussi le fonctionnement du i2c et des autres technologies à disposition que nous pouvons utiliser. Mais aussi différents langages comme le micropython. Il nous as aussi redirigé vers le travail d’un de ses collègues qui travaille sur un produit (raspiaudio) qui pourrait nous intéresser à l’avenir.

3) Radio, lecture de signaux (RDS) & Gestion des fichiers audio

a) Pourquoi le RDS

Le RDS permet d’envoyer des données numériques en même temps que les signaux radios. Ces signaux RDS peuvent contenir des messages, le nom de la station écoutée, ou encore des flash routiers comme avec les TMC (Traffic Message Channel).

Ce Rds peut donc de façon simple et sans interférer avec le signal radio FM nous fournir de l’informations pour notre assistant vocal qui peut servir notre client et notamment les flash routier qui sont très utiles ou encore le nom de la radio qui pourrait permettre de reconnaître la radio et possiblement si oui ou non notre réseau de neurone pourrait se nourrir d’informations en écoutant cette station. Mais aussi l’heure renvoyée peut tout à fait être utile et utilisée pour notre utilisateur.

Nous avons eu des problèmes avec notre module tea5767 pour capter et utiliser le RDS car peu de programme sous python existe. Et la correspondance des librairies n'a pas fonctionné. Il faut utiliser le module SAA6588 pour pouvoir exploiter la possibilité du RDS..

b) Gestion des fichiers audio

Il était nécessaire de pouvoir gérer les fichiers audio pour que l’utilisateur se retrouve avec un fichier prêt à l’emploie (notamment sur l’enregistrement de la radio), mais aussi que le programme ait la possibilité de gérer les fichiers temporaires. Il nous fallait donc différent type de fonctionnalités :

- Créer un audio avec un enregistrement de l'entrée micro.
  Grâce à la librairie sound device on enregistre un fichier en .wav, et on le convertit pour le rendre utilisable pour le speech recognition.
- Fusionner des fichiers audio.

Grâce au bibliothèque en interne wave, nous pouvons fusionner les trames des fichiers audio temporaires pour créer un fichier audio final.

- Supprimer les fichiers audio.

  Grâce à la librairie OS nous demandons au système de supprimer le fichier demandé.

4) Assistant vocal

Pour la partie assistant nous avions une grande liberté des fonctionnalités que nous voulions apporter. De nombreux tutoriels sont disponibles pour créer un assistant vocal.

Nous avons donc synthétisé les concepts et outils présentés que nous avons adaptés à nos besoins:

- le Speech Recognition

  Permet de synthétiser un audio en texte. Que se soit en écoute continue ou avec un fichier audio; Il est possible de modifier les paramètres etc.. Seul défaut pour rester pertinent il se doit d'être utilisé en ligne car l'analyse la plus performante est fournie par les services de Google Recognition.

- La librairie pyttsx3

  Permet de retranscrire un texte en audio. Et ainsi générer les réponses orales de l'assistant à partir d'un texte modulaire généré préalablement. Ce qui permet une grande liberté dans le code, faire des réponses près construites ou bien d'en générer (de) à travers une IA.

Par la suite avec les différentes discussions de professionnels comme Mr Lefevre Franck (Créateur de l'assistant vocal pour personnes âgées, barnabé) et Mr Mercouroff (à travailler sur un projet de tv connecté avec reconnaissance de sous titre etc.. http://tivine.com). On nous a expliqué que les fonctionnalités d'un assistant sont forcément en online, car les bases de données sont bien trop grandes et qu' avec un raspberry nos ressources étaient fortement limitées. Nous avons donc décidé d'adapter le modèle aux contraintes actuelles et plutôt que développer des outils déjà existants et relativement performant. Nous avons pris la décision de laisser la question de "hors-ligne" de côté pour la réalisation du Mvp. Nous permettant malgré tout de nous projeter sur la pertinence des fonctionnalités développées et voir par la suite si la contrainte du hors-ligne est réellement incompatible avec ces dernières.

Ainsi nous avons voulu tester l'implémentation, jusque là impossible, de fonctionnalités propre à internet voir si elles apportent ou non une réelle plus valu au produit. (analyseur radio avec spectre, radio fm etc..).

Nous avons donc importés différentes fonctionnalités présentes chez d'autres produits similaires :

- Demander Heures/Date.
- Demander des informations relatives (grâce à Wikipedia).
- Créer une liste de courses ou todo list.
- Mettre de la musique/vidéo youtube.

Il restera alors à faire tester le Mvp auprès du publique cibles d'origine afin de les valider

5) Analyse syntaxique

Notre but lors de ce stage était donc d'avoir un prototypage du produit, et surtout reconnaître le contexte d'un audio pour pouvoir l'enregistrer en local. Ici nous avons pris le contexte de la météo.

Plusieurs pistes ont alors été développé :

- Analyse syntaxique avec BDD:

  Le but était de retranscrire l'audio que nous avons enregistrés avec le module fm pour le transformer en texte que l'on analysera pour reconstruire un bulletin dans une bdd. Le traducteur audio pourrait ensuite restituer l'audio de lui-même. C'est une piste abandonnée car peu fiable et nous pensons que le produit perdrait de son côté "humain", et nous nous sommes demandé s' il n'était pas possible de restituer l'audio fm "brut" qui garderait la voix originale du locuteur.

- Analyse syntaxique avec audio:

  Nous avons donc choisi de réfléchir différemment, si nous ne pouvons extraire les informations alors pourquoi ne pas tout simplement reconnaître quand on parle de la météo puis enregistrer ces audio fm.
  Nous sommes partie de l'hypothèse que si un audio parle avec des mots clé de la météo alors nous sommes dans le sujet. (avec un nombre de mots que l'on souhaite définir comme seuil de reconnaissance)

  Pour cela nous nous sommes basés sur le speech recognition de google, qui va traduire l'audio en texte. Ici au lieu de nous écouter avec la sortie micro celui-ci va prendre la sortie micro du module fm et retranscrire ce qu'il a compris en texte.
  Nous regarderons ensuite si les mots du texte correspondent au lexique de la météo. Si c'est le cas alors nous enregistrons le passage audio.

Néanmoins plusieurs problèmes se dégagent de cette méthode. Il faut pouvoir paralléliser ces différentes méthodes pour les réaliser en même temps (enregistrement et analyse). cf 7- Optimisation ajoutée (parallélisme). Et pour l'audio final nous n'aurons pas un début et une fin exactement sur le passage de la météo mais nous aurons avant/après le moment intéressant des enregistrement qui ne concerne pas la météo. Puisque comme celui fonctionne en passage audio de x temp et donc il peut y avoir des extrait inutile.

- Analyse des spectre audio avec une IA:

  Les deux méthodes du dessus néanmoins on un gros problème par rapport à notre produit. Il nécessite internet. Les autres reconnaissances d'audio existantes ne sont pas assez développées pour reconnaître un nombre élevé de mots.

  Nicolas et Tristan ont donc développé l'idée d'utiliser du Deep learning ou des réseaux de neurones. Ce qui pourrait nous permettre de ne pas avoir besoin d'internet. cf 6- Deep learning, réseau de neurones.

6) Deep learning, réseau de neurones

Le Deep learning:

Suite à nos nombreuses difficultées à extraire des informations "structurées" du flux radio M.Lefevre nous a redirigé vers M.Mercouroff qui avait réalisé un projet similaire au nôtre. Il est ressorti de cet entretien que le deep learning était une solution plus que pertinente à même de répondre aux contraintes citées ultérieurement. Par la suite Mme Lambert Natacha nous a confortés dans cette idée et nous a même redirigé vers TensorFlow au même titre que M.Mercouroff.

Qu'est-ce que le deep learning ?

Le *deep learning* ou apprentissage profond est un type d'intelligence artificielle dérivé du *machine learning* (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à une IA conventionnelle qui se contente d'exécuter à la lettre des règles prédéterminées. Il repose sur une base de données largement fournie qui déterminera la qualité de l'algorithme généré par le modèle.

Ainsi le format des données, leurs pertinence et leurs diversités sont d'une importance capitale. Toute la capacité d'un algorithme est définie par sa base de données. Mais le secret de cette prouesse repose en grande partie sur les algorithmes.

Dans le cas de la reconnaissance visuelle, pour être performant, doit être capable d'identifier en toutes circonstances des objets divers et sous tous angles.
Ainsi, il sera capable de détecter une voiture sur la route au milieu du paysage. Ceci n'est possible que si la machine a suivi un entraînement poussé. Et ceci passe par la visualisation de milliers de photographies sur lesquelles apparaissent une voiture, de toutes les formes et dans tous les angles possibles.

Lorsque la nouvelle image apparaît, elle est envoyée au réseau de neurones qui se charge de les analyser et de déterminer si l'objet au milieu du cliché est bel et bien une voiture. La machine a gagné son pari ? Elle garde sa bonne réponse au chaud, car elle l'aidera à résoudre d'autres situations similaires le jour où elle devra reconnaître une autre voiture.

Effectivement cela nous a permis de mieux comprendre les mécanismes d'apprentissage qu'une machine peut avoir cependant notre niveau de compétence n'étant pas assez élevé technique d'apprentissage nous en avons discuté avec notre maître de stage qui nous a redirigé vers sa fille Mme Lambert Natacha. Elle nous a expliqué qu'elle avait utilisé le deep-learning pour sa thèse qui était basée sur les cellules cancéreuses à identifier sur radiographie.
Après en avoir discuté avec elle, nous nous sommes orientés vers les réseaux de neurones qui n'est autre qu'une méthode de deep-learning. Il a donc fallu se renseigner sur cette technique d'apprentissage particulière avant de commencer quelques tests.

Réseau de neurone:

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides, et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémenter, et des informations d'entrée au raisonnement logique formel.

En modélisation des circuits biologiques, ils permettent de tester quelques hypothèses fonctionnelles issues de la neurophysiologie, ou encore les conséquences de ces hypothèses pour les comparer au réel.La notion d'*apprentissage*, bien que connue déjà depuis Sumer, n'est pas modélisable dans le cadre de la logique déductive : celle-ci en effet procède à partir de connaissances déjà établies dont on tire des connaissances dérivées. Or il s'agit ici de la démarche inverse : par observations limitées, tirer des généralisations plausibles : c'est un procédé par induction.

La notion d'apprentissage recouvre deux réalités souvent traitées de façon successive :

mémorisation : le fait d'assimiler sous une forme dense des exemples éventuellement nombreux.

généralisation : le fait d'être capable, grâce aux exemples appris, de traiter des exemples distincts, encore non rencontrés, mais similaires.

Pendant nos recherches sur les réseaux de neurones, nous sommes tombés sur une librairie Python qui permettait d'en créer, elle s'appelle : TensorFlow.

TensorFlow:

TensorFlow est une librairie python qui simplifie grandement le développement de réseaux de neurones artificiels. Développée par google et mise à disposition des utilisateurs librement elle vise surtout à simplifier et accélérer la conception d'un réseau de neurones.

Il est possible de récupérer des bases de données via Tensorflow, bien qu'ici cette propriété ne nous soit pas profitable car notre sujet s'éloigne trop des bases de données mis à disposition. Ce qui n'empêche pas derrière de fabriqué soit même une base certe limité mais en accord avec l'optique souhaité.

Création du modèle

Il nous faut tout d'abord créer un modèle qui sera la base de notre réseau de neurones:

Il faut commencer par modifier l'image pour être que la taille de l'image soit celle que l'on veut, une fois les pixels aplatis, le réseau se compose d'une séquence de trois couches **tf.keras.layers.Dense**. Ce sont des couches neuronales densément connectées ou entièrement connectées. La première couche Dense à 256 nœuds (ou neurones) qui possède 28 par 28 connexions soit 789 poids qui seront associés à l'image.

La deuxième couche contient un autre layer avec 128 neurones qui sont tous connectés à la couche précédente ils ont donc tous une liaison avec chaque neurones de la couche précédente.

La troisième et dernière couche couche renvoie un tableau logits d'une longueur de 10. Chaque nœud contient un score qui indique que l'image actuelle appartient à l'une des 10 classes.

Pourquoi trois couches? Le nombre de couches et choisis en fonction de l'image à traiter

En résumé, les données se propagent d'une couche à l'autre où chaque neurone filtre plus ou moins de manière unitaire le résultat et active des sorties. L'apprentissage se présente alors sous la forme d'une rétroaction qui vise à moduler automatiquement, par le modèle, le poids de chaque embranchement afin de faire concorder le résultat réel avec le résultat effectif.

Si prédiction il y a, probabilité également et c'est comme cela que les neurones reconnaissent les images comme faisant partie de leur classe ou non. Il y a donc des probabilités pour chaque sortie.

Apprentissage du modèle:

On va utiliser une méthode pour nous permettre de définir une fonction d'erreur (loss), un optimizer ce qui nous permettra de réduire l'erreur précédente, enfin un metrics qui est le pourcentage de réussite des prédictions:

Avant que le modèle ne soit prêt pour l'entraînement, il a besoin de quelques paramètres supplémentaires. Ceux-ci sont ajoutés lors de l'étape de compilation du modèle:

1-Fonction de perte : Cela mesure la précision du modèle pendant l'entraînement. Vous voulez minimiser cette fonction pour "diriger" le modèle dans la bonne direction.

2-Optimiseur: c'est ainsi que le modèle est mis à jour en fonction des données qu'il voit et de sa fonction de perte.

3-Metrics :Utilisé pour suivre la formation et les étapes d' essai. L'exemple suivant utilise la *précision* , la fraction des images correctement classées.

Ce qui nous permet ensuite de lancer l'entraînement du modèle en fonction du nombre “d’époques” ou d’itération que nous voulons.

Après avoir compris comment utiliser TensorFlow, il nous restait une zone d’ombre qui nous empêchait d’avancer, car rappelons le ce n’est pas les images que nous voulons traité mais des bandes son, il a donc fallut trouver une solution pour régler ce problème.

Le lien avec notre projet

Plusieurs possibilités se sont offertes à nous lors de nos recherches.

Premièrement, l’optique choisie fut évidemment de donner au réseau de neurones un fichier audio. Or le model que l’on a vu lors de nos recherches ainsi que nos tests avaient les particularités suivantes, l’image était binarisée (noir et blanc) car en trois dimensions à l’origine, mais elle était également pixélisée afin de réduire le nombre de pixel de cette dernière et enfin elle était aplatie pour qu’il y est à chaque entrée du réseau un seul est unique pixel.

Nous avons donc cherché à adapter ce procédé à un fichier audio. Le premier constat était qu’à la différence de l’image, l’audio ne comporte que deux dimensions, un temps et une fréquence . La première étape était donc anecdotique, sauf qu’un problème plus ardu nous fit douter, le format audio en lui-même.

Le format audio

En python le format le plus accessible est le .wav or ce type de fichier est non seulement volumineux mais aussi des informations essentiels comme l’encodage mono ou

stéréo n’y sont pas clairement visibles/dissociables, et ainsi il faut être regardant sur chaque fichier utilisé.

Pour régler le problème de de volume, qui vise ainsi à réduire drastiquement le nombres de données à analyser, nous nous sommes tourné vers d’autre format tel que le “Mp3”,”Ma4”,...
Mais rien de très concluant à utilisé avec python. La plupart des bibliothèques déjà existantes sont spécialisées sur le “.wav”, rendant difficile les transformations suivantes.

Analyse Wav

De retour sur le format .wav, on s'est attelé à transformer la donnée audio. Pour ce faire nous avons émis plusieurs hypothèses à travers le biais de représentation graphique afin d'isoler des particularités propre à un fichier audio, ou plutôt un son, le spectrogramme, représentation spectrale et même temporelle.
Il en est ressorti que le spectrogramme est le plus à même de mettre en évidence les propriétés des mots par l’isolation de syllabes et ainsi permettre une identification audio.
La conjugaison de ces représentations peuvent donner “une carte” d’identité plus que pertinente à un fichier audio.

Détecter un Jingle

Seulement la propriété de la voix, tessiture, l’accent et l’ accentuations (le français étant moin impacté par ce paramètre que d’autre langue ils peuvent tout de même modifier la fréquence d’une syllabe), ne permettent pas l'entraînement d’un modèles pertinent à même d’être transposé à n’importe quel locuteur sans une base de donnée extrêmement fournis, diversifiée et structurée.

Nous nous sommes donc rabattus sur la reconnaissance de “jingle” permettant d’identifier de manière relativement sûre le programme en cours, amorcé son enregistrement, on peut ainsi analysé plus en détails avec le modèle actuel du projet (connecter à internet) la partie réellement pertinente. Pour entamer l’apprentissage et surtout la validation d’un modèle à même de remplir ce rôle, deux éléments sont ressortis.

Le choix du jingle

Tout d’abord le jingle en lui même, toutes les radios n’utilise pas forcément de jingles avant leurs bulletins radio, routier … ce qui rend le choix de la radio plus important (dans une optique de disponibilité) il s’avère que France Inter utilise des jingles normalisés en fonction de la période de la journée, et offre par la même occasion des informations non-négligeables supplémentaires. Sans oublier que cette radio est disponible sur une grande partie du territoire car c’est une radio d’échelle nationale.

Par la suite l’important était d’identifier le jingle de manière fiable et ce peut importe la qualité de la réception. Plutôt que d’enregistrer la radio quotidiennement afin de constituer un corpus suffisant, il a été décidé de simplement modifier le fichier original en le détériorant, de manière graduée. Que ce soit en hachant l’audio, en le modulant, en ajoutant du bruit par dessus ect.

Ainsi un corpus de trentes fichiers différents ont été créés comportant chacun 4 jingles distincts. Ce corpus n’a été réalisé que pour valider la pertinence du réseau de neurones pour analyser à notre niveau l’audio sans passer par un intermédiaire type speechToTexte qui requiert nécessairement internet. Et de la possible extension du modèle à autre chose que des jingles, des phrases clé ou autre.

7) Optimisation ajoutée

a) Problèmes & résolutions

Notre premier problème a été de ne pas pouvoir implémenter et régler un réseau de neurones pour enregistrer notre passage à la radio quand le sujet était la météo(manque de temps et de connaissances). Nous avons donc dû nous pencher sur un substitut. Nous enregistrions donc une entrée micro de x secondes pour ensuite grâce à un speech to text pouvoir définir si le lexique utilisé dans ce bout était du registre de la météo.

Vient alors un problème de taille: comment allons nous enregistrer un flux audio continu tout en l’analysant de manière efficace sans pour autant stocker des centaines de fichiers son. Nous avons alors opté pour une stratégie d’analyse de son en parallèle de l’enregistrement du flux audio qui suit. Ce qui n’a pas été très concluant car nous n'utilisons pas de threads. Donc pour régler ce problème nous sommes partis dans l’optique d’extraire la donnée d’un fichier son avec le Speech To Text pour ensuite définir si oui ou non nous la gardons comme faisant partie du bulletin météo. Pour cela nous avions besoin d’utiliser du parallélisme et plusieurs Threads pour pouvoir au moins enregistrer un flux audio en continu mais aussi vérifier le contexte de cet audio pour l’ajouter en temps que partie du bulletin météo qui pourra être restitué à la demande de l’utilisateur.

b) Gestion des fichiers

La gestion de nos fichiers audio est assez simple nous gardons seulement les fichiers qui sont captés comme étant de la météo sur l’instant sinon le fichier est supprimé. Si nous avons plusieurs (au moins deux) fichiers captés comme étant de la météo, alors nous fusionnons ces deux fichiers. Ensuite on stocke ce fichier comme étant le dernier bulletin météo.

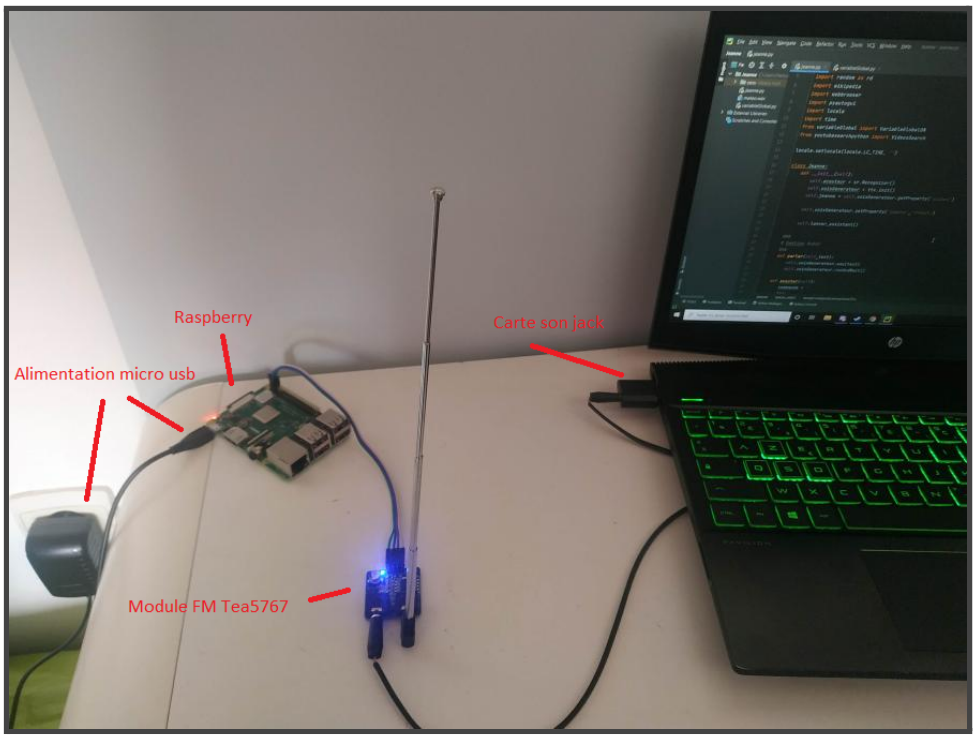
8) Etat du produit au rendu de stage

Sur ordinateur:

Travail à partir du pc sans RDS avec de simple fichier audio, faire l'analyse de ces derniers (syntaxique et autres) et montrer en parallèle l'assistant vocal sous son meilleur jour.

Sur Rasberry:

Utilisation du module fm pour récupérer le flux radio dont nous avons besoin, toutes nos librairies utilisées pour notre programme sont installées dessus et cela fonctionne correctement.



9) Sources

Trello :
https://trello.com/invite/b/BDyeYckm/cd1785b3471a2da5d89eca7d087bb2ea/jeanne-todo-list

tuto how ADC work:
https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters

what is dsp:
https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html#

Module FM:

https://www.piddlerintheroot.com/fm-radio-tea5767/
pistes
https://www.youtube.com/watch?v=yWf9uxL6zgE&ab_channel=RalphSBacon
https://www.allaboutcircuits.com/projects/build-an-arduino-controlled-am-fm-sw-radio/
https://www.silabs.com/documents/public/data-sheets/Si4840-44-A10.pdf

Assistant Vocal:
https://youtu.be/3YzTTj7qM9k
https://www.youtube.com/watch?v=bU4Dcq1iF2A&ab_channel=FORMASYSFORMASYS
https://www.youtube.com/watch?v=A9_0OgW1LZU&ab_channel=NicholasRenotte

Gestion de l'audio:
https://stackoverflow.com/questions/2890703/how-to-join-two-wav-files-using-python

Natural Language :
https://www.nltk.org/
https://spacy.io/models/fr

Réseau de neurone :
https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition
https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels
https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/
https://www.tensorflow.org/tutorials/keras/classification?hl=fr

Parallélisme :
Emmanuel Cagniot
https://www.youtube.com/watch?v=fKl2JW_qrso&ab_channel=CoreySchafer
https://www.youtube.com/watch?v=vdjZvxAI5d4&ab_channel=FormationVid%C3%A9o

Librairies utilisé :
https://pypi.org/project/wikipedia/
https://www.youtube.com/watch?v=hP7Ac8S9Tgs&list=PLpEPgC7cUJ4byTM5kGA0Te1jUeNwbSgfd

https://doc.ubuntu-fr.org/espeak

https://raspiaudio.com/