DeepLanguage

Rapport de projet informatique de 2ème année à l'ENSC

Réalisé par Joseph Beasse

Sommaire

Contextualisation 3
Travaux réalisés 4
1. Sélection des données 4
2. Pré-traitement des données 4
3. Sélection des “features” (caractéristiques) 5
a. Spectrogrammes 5
b. MFCCs 6
4. Traitement des données 6
a) Calcul des coefficients cepstraux de Mel 6
b) Préparation des datasets 7
5. Choix de l’architecture du modèle 8
6. Définition du modèle 8
7. Validation des performances 10
8. Création du site web (front/back) 11
Problèmes rencontrés 12
1. Sensibilité du modèle 12
2. Upload du site web 13
Planning de travail 13
Pistes d’évolutions 15
Bilan personnel 15

Contextualisation

L’intelligence artificielle possède de nombreuses applications dans la vie courante. Certaines sont discrètes mais ont pour autant été des révolutions. Le projet DeepLanguage a pour objectif de développer une intelligence artificielle capable d'identifier les langues parlées.

Cette initiative est motivée par l'utilisation croissante des assistants vocaux tels que Siri et Alexa, ainsi que la nécessité de transcrire des textes dans différentes langues. Les traducteurs fournis par de grandes entreprises comme Google sont également équipés d’algorithmes permettant d’identifier la langue lors d’une retranscription.

Assistants audios d'Apple (Siri) et Amazon (Alexa)

La langue est une forme complexe de communication qui est codée dans les sons produits par notre voix. Les humains sont capables de reconnaître facilement une langue qu'ils ont apprise en écoutant ces sons. Cependant, la question qui se pose est de savoir comment enseigner à un réseau de neurones artificiels de reconnaître ces langues en utilisant des données numériques.

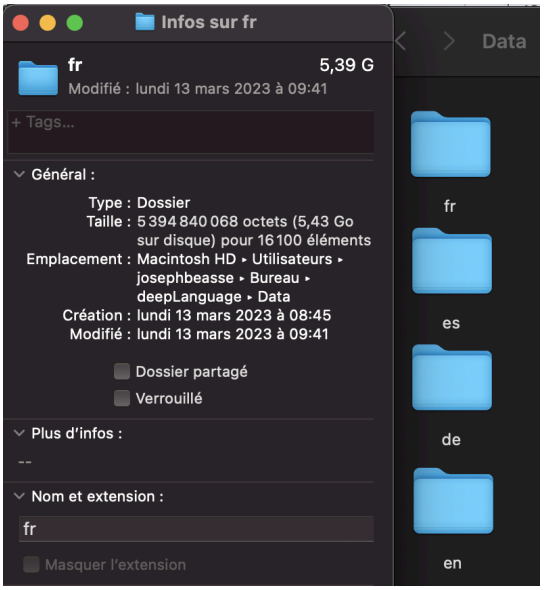
Il est important de situer les limites d’un projet avant de le commencer. L'identification de la langue parlée n'est pas un problème trivial pour les machines car cela nécessite une quantité considérable de données pour enseigner à ces dernières à distinguer chaque petit extrait d'une langue. Cela devient de plus en plus difficile lorsqu'il y a plusieurs langues à distinguer et que différents accents, âges, sexes et autres traits influencent la voix humaine.

Travaux réalisés

1. Sélection des données

La première étape de tout projet en apprentissage automatique (AA) ou profond est l'acquisition de données. Sans données, pas de projet.
DeepLanguage a pour but d'identifier la langue parlée dans un audio court. Il faut alors trouver ou créer un dataset contenant beaucoup de fichiers audio courts dans les langues choisies (ici français, anglais, allemand et espagnol) équitablement répartis. Ne disposant pas d'assez de temps et de moyens j'ai utilisé la banque d'audios fournie par Mozilla et appelée Common Voice. Ce dataset est une banque d'audio au format mp3 conçue par les visiteurs du site associé. Ces derniers doivent lire une phrase montrée sur leur écran. Pour s'assurer de leur bonne lecture, les audios ne seront par la suite publiés que s'ils ont été validés par d'autres visiteurs.

Par langue, 16 100 fichiers audios variant entre 4 et 7 secondes ont été récupérés et placés dans des dossiers nommés par les initiales de leur langue facilitant leur future labellisation (étiquetage).



Informations sur le dossier "fr" du parent Data

2. Pré-traitement des données

Une fois les fichiers audios récupérés et séparés selon la langue, il faut s'assurer de bonnes pratiques courantes en IA. Le dataset doit contenir des échantillons variés, équitablement distribués et dans un format pratique d'utilisation. Le langage de programmation le plus répandu pour réaliser des algorithmes d'apprentissage automatique est Python. C'est ce langage qui va également réaliser tout le traitement des données du projet. Le tableau ci-dessous recense les principaux scripts utilisés pour le pré-traitement des données.

Nom	Description
towav	Permet de convertir l'intégralité des fichiers audios au format mp3 d'un dossier vers le format wav qui est non compressé. Ce qui signifie que les données audio sont stockées sous forme brute sans perte de qualité.
notawavfile	Permet de lister tous les noms de fichier qui ne sont pas des wav ou qui ont mal été convertis.
findNan	Repère si un fichier audio est corrompu (erreur dans le traitement ou la conversion), et si ce dernier possède une valeur NotANumber.
remove	Supprime tous les fichiers d'un dossier qui ne sont pas d'extension .wav
fileSelector & fileTransfer	Chaque dataset (4 pour les 4 langues) disposaient d'un CSV mentionnant quels audios avaient été vérifiés et combien de fois. Ces scripts permettent d'enlever ceux vérifiés et faux, et ceux non vérifiés.

3. Sélection des "features" (caractéristiques)

Une des étapes les plus importantes pour un projet d'apprentissage automatique est la sélection de "features" ou caractéristiques permettant de décrire le modèle. En se posant les bonnes questions il est généralement plus simple de sélectionner les bonnes caractéristiques permettant au mieux de décrire ce que l'on recherche.
Comment représente-on un signal sonore?
Comment la voix est-elle représentée ?
Si elle existe, quelle information représente au mieux la langue dans un signal sonore ?

a. Spectrogrammes

Les spectrogrammes ont été utilisés pendant longtemps pour la tâche d'identification de langues en traitement de signal. Ils représentent graphiquement l'évolution de la fréquence d'un signal sonore en fonction du temps.
Malheureusement, un signal de faible qualité ou d'une durée d'enregistrement courte entraîne une mauvaise représentation des caractéristiques.
Il faut alors trouver une représentation plus minutieuse de la voix et qui est moins sensible.

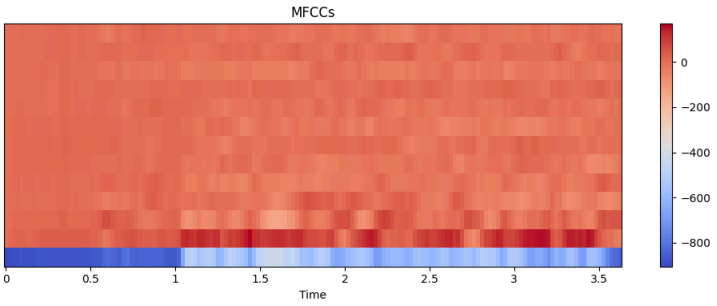
b. MFCCs

Pour remédier à ces problèmes, les coefficients Cepstraux de fréquence mel (MFCC) ont été introduits. Les MFCC sont une représentation compacte des caractéristiques du spectre de puissance d'un signal sonore et sont plus efficaces pour l'identification de langues que les spectrogrammes en raison de leur résistance aux variations de volume et de tonalité.
Que sont exactement ces coefficients ?
Dans le domaine de la reconnaissance vocale, on utilise un outil appelé les coefficients cepstraux pour comprendre les sons que produit la voix. Les coefficients cepstraux contiennent des informations sur la façon dont les sons de la voix sont créés. Ils permettent de séparer les sons qui viennent de la gorge (les cordes vocales) et ceux qui viennent de la bouche (le tractus vocal) parce qu'ils sont situés dans des parties différentes du graphique cepstral. Si un coefficient cepstral a une valeur positive, cela signifie que le son est plutôt doux ou musical, tandis qu'un coefficient cepstral négatif indique plutôt un son plus bruyant ou sifflant. Les coefficients cepstraux les plus bas contiennent des informations sur la forme globale du son, tandis que les coefficients plus élevés donnent des détails plus précis.
Dans la littérature, on utilise entre 12 et 20 coefficients cepstraux pour analyser les sons de la parole, en utiliser trop pourrait rendre l'entraînement de modèle très complexe.

4. Traitement des données

a) Calcul des coefficients cepstraux de Mel

En apprentissage automatique, le nombre de neurones à l'entrée du réseau est défini par la taille des données. Il est donc nécessaire de traiter chaque fichier audio avec le même nombre de données. Arbitrairement, le choix pris pour cette étude est de remplir ou tronquer chaque audio pour que ces derniers durent 5 secondes.
Dans un premier temps, le choix de la librairie de traitement des audios était l'API TensorFlow, à l'aide de son multithreading et son optimisation il s'agissait d'un choix cohérent pour traiter de grandes quantités de fichiers. L'api est très bien documentée et permet une implémentation et un calcul rapide sur l'ensemble du corpus d'audios. Cependant, j'ai finalement choisi d'utiliser une bibliothèque dédiée au traitement sonore pour améliorer la qualité de mes résultats et simplifier au maximum le code: Librosa



MFCCs d'un signal audio (anglais) avec librosa.

Une fois calculée, la taille d'entrée de notre réseau de neurones est de 13x157. 13 représente le nombre de coefficients cepstraux choisis pour représenter au mieux la voix humaine. Les 157 correspondent à de petites fenêtres temporelles de 32 millisecondes environ.

b) Préparation des datasets

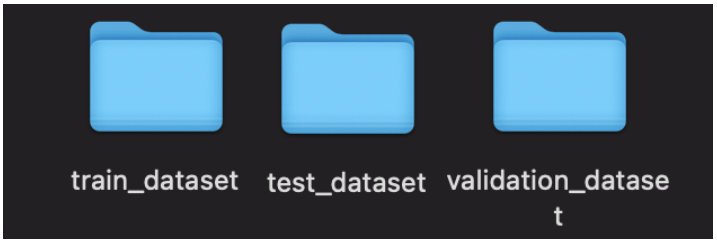
Pour réaliser un apprentissage supervisé, il faut pouvoir montrer à l'algorithme le vrai du faux. Pour cela, chaque matrice calculée précédemment doit être augmentée avec l'information lui indiquant la langue en question.
L'encodage de la langue est fait numériquement à l'aide de la méthode "One-Hot". Il s'agit d'une représentation d'une information qualitative sous la forme d'un vecteur de binaires. 1 si la langue est celle du bon indice, 0 sinon.

One-Hot Encoding

Langue	Français	Espagnol	Allemand	Anglais
Français →	1	0	0	0
Allemand	0	0	1	0
Espagnol	0	1	0	0
Anglais	0	0	0	1

Exemple d'encodage en vecteur de 0 ou de 1 dit "One-Hot"

Une fois les données traitées, ces dernières sont séparées aléatoirement en 3 dossiers. Un d'entraînement comprenant 75% du corpus, un de validation et un de test comprenant 12,5% chacun du corpus de valeurs.



Données sauvegardées sous 3 dossiers.

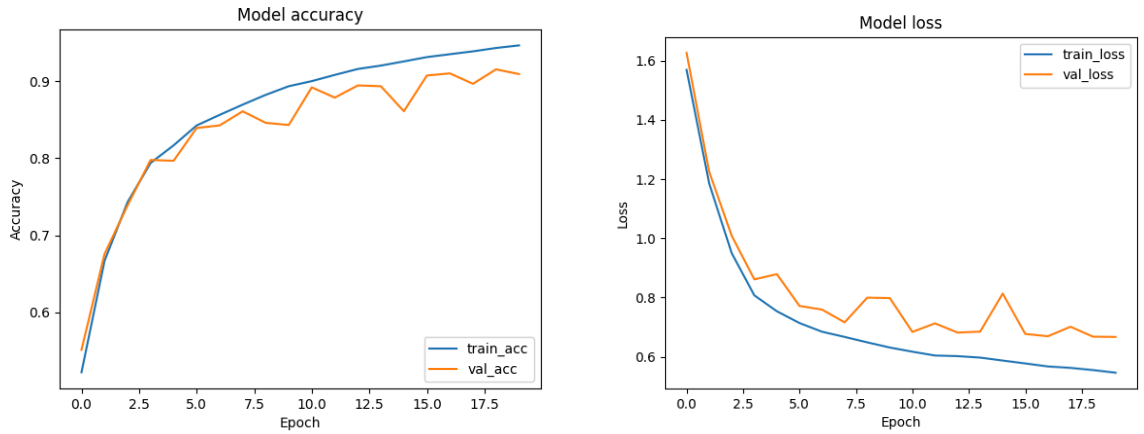
5. Choix de l'architecture du modèle

D'après la littérature, le type d'algorithme majoritairement utilisé pour réaliser de l'identification de langue est le réseau de neurones à convolution (CNN).
Les CNN sont conçus pour traiter des données structurées telles que des images, mais ils peuvent également être adaptés pour le traitement de signaux sonores.
Les couches de convolution dans un CNN sont capables d'extraire des caractéristiques telles que les transitions fréquentielles et temporelles, tandis que les couches de regroupement (max pooling) peuvent réduire la dimensionnalité de ces caractéristiques pour une classification plus précise.
Une autre architecture viable pour cette tâche est le RNN.
Ce dernier peut maintenir dans une mémoire à long terme des informations précédentes d'une séquence, ce qui peut être utile pour la classification de la langue.
Plusieurs modèles ont été entraînés afin d'évaluer lequel à les meilleures performances. Les résultats seront détaillés par la suite.

6. Définition du modèle

Itérativement, plusieurs modèles ont vu le jour. Certains étaient trop volumineux, d'autres étaient victimes du phénomène de sur-apprentissage dû à leur trop grande complexité. Le modèle qui a le plus performé dispose de 16 millions de paramètres.
C'est un CNN séquentiel élaboré pour prévenir du phénomène d'overfitting.
Le modèle comprend des couches de convolution pour extraire les caractéristiques du signal sonore, des couches de mise en commun pour réduire la dimensionnalité, des couches de normalisation pour accélérer l'apprentissage et des couches entièrement connectées pour la classification. Le modèle a une couche de sortie

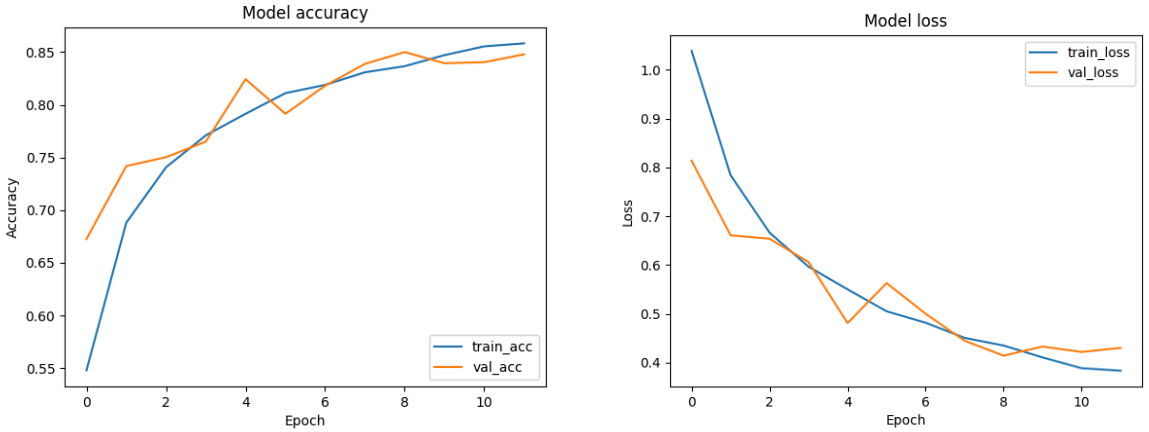
avec une fonction d'activation softmax pour prédire la probabilité d'appartenance à l'une des quatre langues.
Il utilise une fonction de perte catégorielle et un optimiseur Adam pour l'apprentissage. Enfin, il est entraîné sur des données d'apprentissage et de validation pendant 20 époques (epochs) avec un arrêt anticipé pour éviter le sur-apprentissage.
Voici les performances obtenus lors de l'apprentissage de ce modèle appelé "Modèle 3"



Précision et perte d'entraînement et de validation du modèle 3 en fonction des époques.

Sur ces courbes, on remarque que la courbe de validation suit la même forme que celle d'entraînement, cependant, des fluctuations persistent et un léger décalage se prononce au bout d'une dizaine d'époques. Cela témoigne donc d'un apprentissage sur-apprenant sur les données présentées à l'entraînement, et une difficulté à généraliser à un nouveau corpus encore jamais étudié.

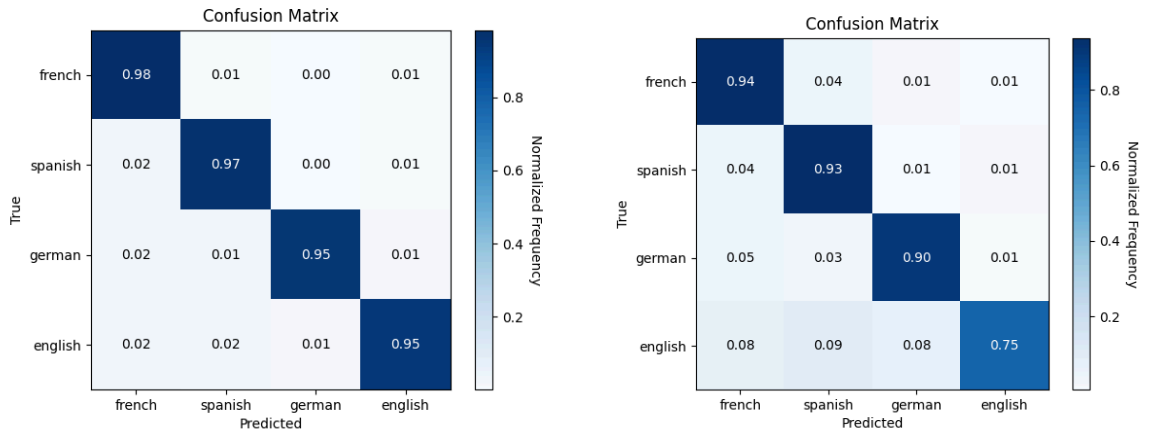
À titre comparatif, le modèle récurrent offre des performances moins concluantes en utilisant des couches bidirectionnelles de mémoire à long terme et de plus la phase d'entraînement est 2 fois plus longue. Comme le montre les figures ci-dessous, le modèle apprend plus vite mais oscille autour de la courbe d'entraînement. En optimisant les hyperparamètres et en ajoutant quelques dizaines d'époques supplémentaires, le modèle RNN pourrait se montrer très performant en raison de sa rapide convergence et de ses valeurs de pertes peu élevées.



Précision et perte d'entraînement et de validation du modèle RNN multicouche en fonction des époques.

7. Validation des performances

Pour valider les résultats, j'ai mis de côté, comme cité précédemment, un dataset de test. Ce dernier contient 8050 fichiers audios soit environ 2000 par langue. Ces derniers n'ont pas été utilisés pour l'entraînement du modèle ni pour l'étape de validation du modèle. Afin d'avoir un résultat précis et facile à analyser, l'évaluation des performances est réalisée par la construction de la matrice de confusion.
La matrice de confusion montre les pourcentages de vrais/faux positifs/négatifs.
Comment lire la matrice ?
Sur chaque ligne est présentée tous les fichiers audios étant réellement dans la langue correspondante, tandis qu'en colonne sont représentées toutes les prédictions effectuées par le modèle.

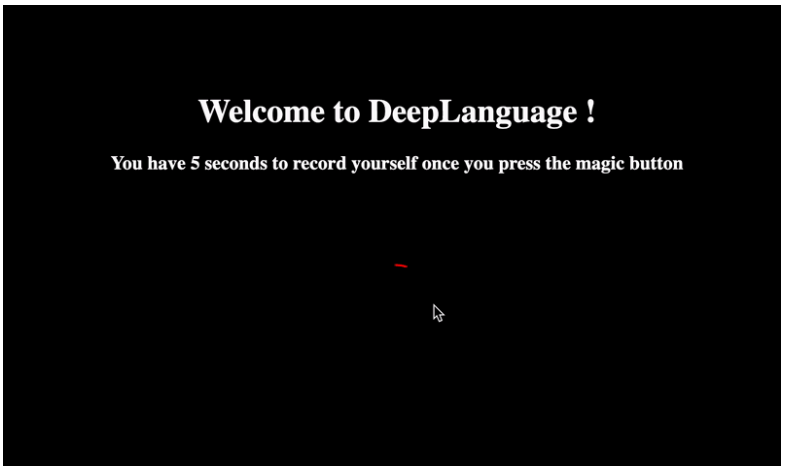


Matrices de confusion des modèles 3 à gauche et RNN multicouches à droite.

Par exemple, le modèle 3 ne prédit jamais d'allemand lorsque l'audio est de l'espagnol. Il réussit également 95% de ces prédictions lorsqu'il s'agit d'anglais. Ces résultats semblent assez élevés pour un simple CNN. L'objectif était d'atteindre 85%, l'écart positif constaté peut venir du fait qu'une personne à tendance à soumettre plusieurs audios (et donc de la même langue) sur le site de Common Voices. Lors de la séparation aléatoires, des voix déjà rencontrées peuvent alors être testées par le modèle. Il faudrait élaborer un datasets de tests plus diversifié afin d'obtenir le résultat escompté.

8. Création du site web (front/back)

Afin de pouvoir utiliser l'algorithme sur des cas concrets, un site web a été réalisé. Ce dernier est codé en Flask afin de disposer de toutes les librairies de Python.
Le site dispose d'un simple bouton "record" sur la page d'accueil. La prédiction est ensuite affichée après les 5 secondes d'enregistrement audio.
Le front du site est réalisé en HTML, CSS et Javascript.
Le back lui, est entièrement conçu en Python.
Pour des raisons cités plus loin dans le rapport, une autre version du bac a été développée côté client en Javascript, elle réside sur une autre branche du projet. Ce projet se nomme DeepLanguageWebsite.



(Gif) Le site web de DeepLanguage

Problèmes rencontrés

1. Sensibilité du modèle

Les différentes versions du modèle montrent qu'il est très sensible aux perturbations. Ces dernières proviennent en partie du jeu de données avec lequel il est entraîné. En effet, ce dernier se base sur de la lecture de texte à voix haute et non de dialogues, plus courant dans la vraie vie. Il se base essentiellement sur des personnes natives dans leur langue. Une utilisation dans un contexte de dialogue ou discours témoigne de grands écarts dans les résultats.

Plusieurs alternatives ont été essayées pour parvenir à ce problème.
Parmis celles ci on compte:

a) Un script pour les audios longs

Pour tester le modèle sur des dialogues plus longs que 5min, il fallait alors élaborer un processus permettant de couper l'audio en portions de 5 secondes et de réaliser une prédiction sur chacun d'eux. Malheureusement certains audios commencent au milieu d'une phrase, finissent par un soupir. La qualité des audios s'éloignait en effet beaucoup de celle avec laquelle le modèle à été entraîné. La précision du modèle obtenu pour 10 dialogues de 3 minutes chacun pour chaque langue est de 48%.

b) Des améliorations dans les modèles déjà existants

Il existe de nombreux moyens de réduire le phénomène de sur-apprentissage dans les modèles. Des méthodes de régularisation, de callback et de dropout ont été implémentés afin de réduire ce phénomène. Il est important de noter que ces méthodes ralentissent l'apprentissage et peuvent dans certains cas empêcher ce dernier de converger.

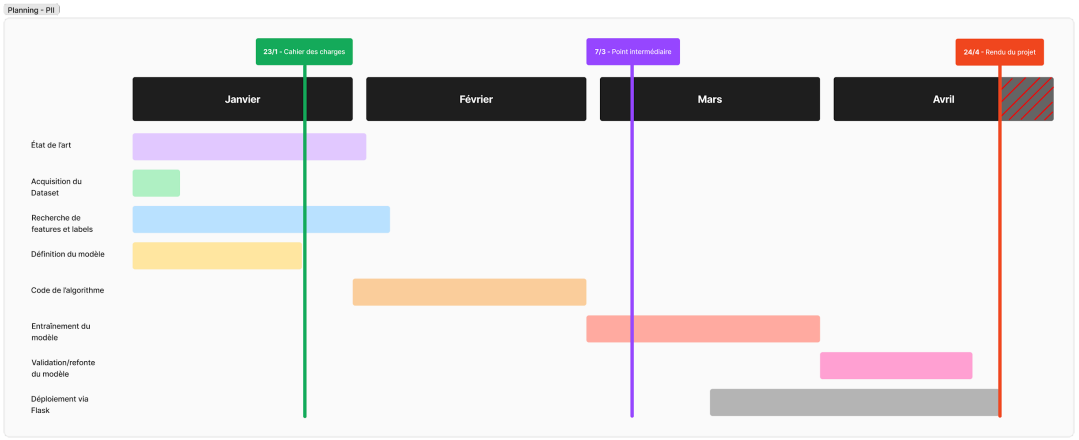
c) Augmentation de données

La méthode la plus connue et la plus utilisée pour résoudre les problèmes de sensibilité d'un modèle est l'augmentation de données.
Cette méthode consiste en une série de (petites) modifications des données d'entraînement afin d'en générer de nouvelles légèrement différentes. Pour des images, cela consiste par exemple à la retourner. Une photo de chat retournée reste une photo de chat. Pour des fichiers audios c'est légèrement différent, on peut faire varier le bruit, la hauteur, la vitesse et toutes les caractéristiques qui définissent un son. Les tests effectués disposaient de valeurs trop élevées de modifications et le modèle ne pouvait converger. Il faudrait alors étudier les caractéristiques de la voix afin d'y connaître les paramètres précis à modifier.

2. Upload du site web

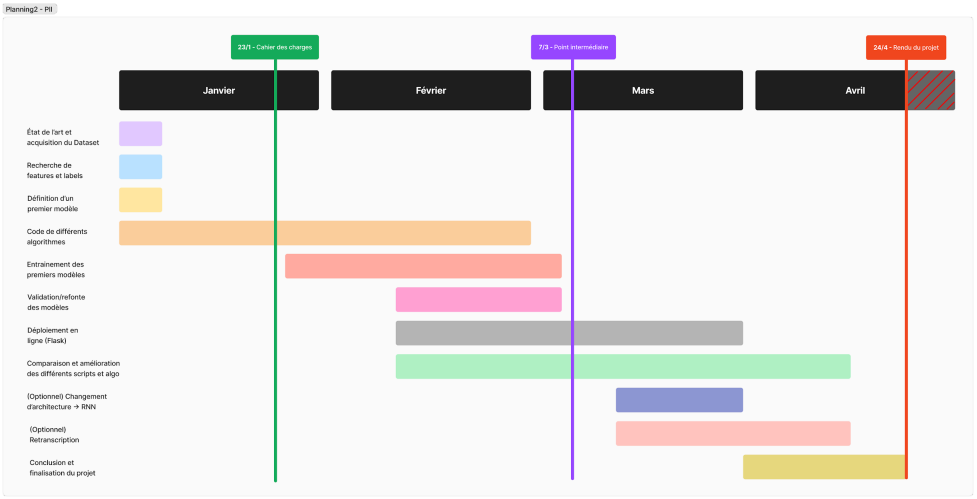
Le traitement de l'audio est réalisé sur Python par le package Pydub qui nécessite une carte audio. Cependant les serveurs distants sur lesquels le projet peut être téléversé ne disposent pas d'une telle carte. Le projet fonctionne parfaitement en local, mais l'enregistrement audio devra obligatoirement passer de serveur à client en utilisant le langage javascript pour traiter l'audio.
Après avoir revu tout le traitement backend d'acquisition de l'audio, l'implémentation en JS a été réussi. Cependant, le souci à présent vient de l'hébergement du site. Certains hébergeurs bloquent les requêtes JS avec lesquelles je fonctionne. D'autres ne permettent pas la possibilité de téléverser des fichiers volumineux comme les modèles TensorFlow sans payer une somme d'argent mensuelle.
Le projet après de nombreux essais infructueux ne reste donc utilisable qu'en local.

Planning de travail



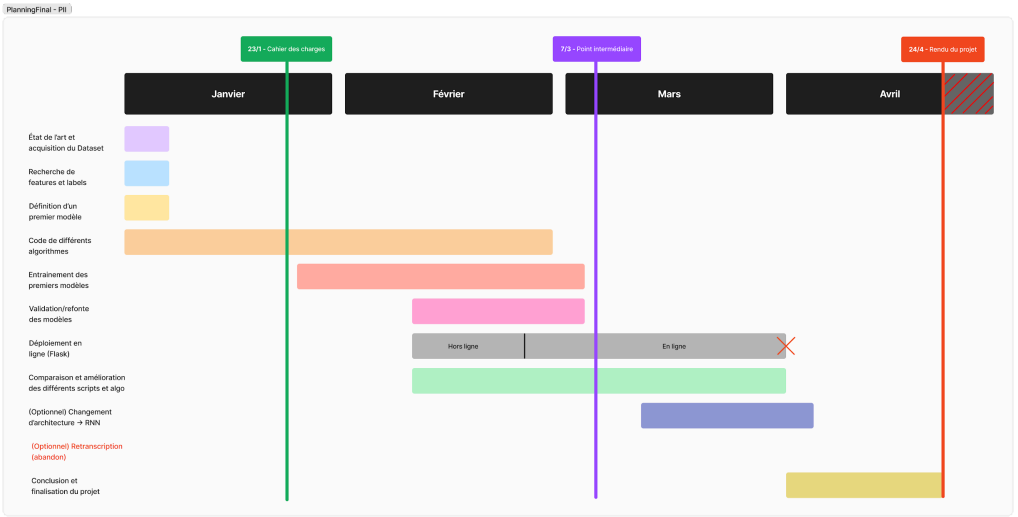
Planning initial de DeepLanguage

L'objectif initial de DeepLanguage était la conception d'un réseau convolutionnel afin d'identifier la langue d'un fichier audio et de fournir ce service à l'aide d'un site web. Des objectifs intermédiaires et une restructuration du planning ont eu lieu après les premières semaines. En effet, j'ai bénéficié de beaucoup de temps libre et de motivation en début de projet pour avancer sur ce projet informatique individuel si bien que les premiers algorithmes implémentant les spectrogrammes sont arrivés courant janvier. Suite à une réunion avec mon tuteur, une restructuration du planning a eu lieu, ce dernier définit de nouveaux objectifs et précise plus en détails les différentes étapes primordiales du projet.



Planning intermédiaire de DeepLanguage

Une fois le planning redéfini, le projet a avancé comme convenu. Pour tenir au courant mon tuteur de l'avancée du projet, un mail chaque semaine, récapitulant les travaux effectués et ce que je devais faire pour la semaine à venir, était envoyé. Enfin, certains problèmes techniques intervenus comme le temps d'entraînement des modèles très élevé (6h tout au moins) et l'impossibilité d'upload le site web ont entraîné des modifications dans le planning comme indiquées dans la version finale ci-dessous.



Planning final de DeepLanguage

Pistes d'évolutions

DeepLanguage est une amorce de projet visant à créer un assistant vocal.
Dans les étapes de ce dernier, on compte la reconnaissance de langue, la transcription de texte et la génération de réponses.
Pour aller plus loin, plusieurs pistes peuvent être abordées. Dans un premier temps, il serait intéressant de réaliser son propre dataset documenté, en y inscrivant l'âge, le sexe, l'accent afin d'y ajouter ces caractéristiques lors de l'entrainement et de peaufiner les performances du modèle. Dans un second temps, le cap des 85% de précision sur le dataset de validation (autre que des données extraites de Common Voice) atteint, augmenter le nombre de langues prises en charge par DeepLanguage permettrait de raffiner le modèle afin qu'il aille chercher plus en détails les véritables éléments discriminants entre 2 langues.
De plus, pouvoir continuer le développement des modèles récurrents en y comprenant précisément les tenant et aboutissants serait une grande avancée vers la deuxième étape du projet, autrement dit, la retranscription.
Enfin, afin de toucher un public plus large et tester l'algorithme dans diverses conditions, le déploiement du site peut devenir une piste prioritaire quant à l'évolution du projet.

Bilan personnel

À titre personnel, je me suis lancé dans l'aventure DeepLanguage afin d'en apprendre plus sur l'intelligence artificielle. C'est un secteur qui me passionne tout particulièrement et dont je souhaiterais développer le plus de compétences possibles. La charge de travail impliquée a été contrebalancée par ma motivation et mon envie d'apprendre de jour en jour. Ce projet m'a beaucoup enseigné sur les projets d'IA que je pourrais rencontrer en société. On ne fait pas de l'IA pour faire de l'IA, d'autres méthodes subviennent parfaitement à certains besoins.
Les données sont le cœur du problème, leur obtention, leur labellisation et l'extraction de caractéristiques importantes permettant de décrire un problème sont la partie la plus importante du projet. Cette expérience semi-professionnelle m'a confortée dans ma volonté de carrière et je souhaite continuer ce projet sur mon temps personnel.
Les outils que j'ai utilisé sont en majeure partie issus de l'API de tensorflow, il est néanmoins important de comprendre comment fonctionne chaque bloc de code que l'on écrit. Lire une documentation scientifique n'est pas un simple outil de débogage mais bien un outil pédagogique permettant d'enseigner. Ce projet m'a également permis de trouver un stage et de faire valoir mes compétences aux yeux des entreprises, son apport a été plus que positif.

Code

Le code pour le projet DeepLanguage est disponible sur
github.com/Zoko91/DeepLanguage
Celui pour DeepLanguageWebsite est disponible sur
github.com/Zoko91/DeepLanguageWebsite

Pour télécharger l'ensemble des documents comprenant les archives, l'environnement de travail et les données utilisées pour l'entraînement du modèle, cliquez sur ce lien: FilesenderLINK