

Institut Supérieur de l'Électronique et du Numérique

Tél. : +33 (0)2.98.03.84.00

Fax : +33 (0)2.98.03.84.10

20, rue Cuirassé Bretagne

CS 42807 - 29228 BREST Cedex 2 - FRANCE

Voice Cloning



Proposé par : Matthieu Saumard

Thématique : Indifferent

LOPEZ Théo

Domaine professionnel :

Cybersécurité

CHARRIER Yohann

Domaine professionnel :

Cybersécurité

SOMMAIRE	2
GLOSSAIRE	4
INTRODUCTION	5
CAHIER DES CHARGES	6
Contexte	6
Quelques contraintes	6
L'Analyse client	7
Cas d'usage	7
DESCRIPTION FONCTIONNELLE	8
1 - Partie Système	8
A - Encoder	8
B - Synthesizer	9
C - Vocoder	9
2 - Partie Interface	10
ANALYSE DU PROJET	11
GESTION DU PROJET	12
DÉROULEMENT DU PROJET	12
1 - Phase de Lancement	12
2 - Domaine professionnel	13
3 - Phase Finale	13
OUTILS DE VALIDATION TEST	14
COMPTE RENDU DE RÉUNION ET REPORTING	15
DÉVELOPPEMENT TECHNIQUE	17
INTERFACE GRAPHIQUE	18
1 - Fonctionnement Bootstrap	19
2 - Fonctionnement Flask	19
3 - Arborescence	20
4 - Caractéristique implémenté	21
A - Le choix de la voix	21
B - Le Choix de l'audio	21
C - Autre Informations	21
BASE DE DONNÉES	22
1 - Encoder :	22
2 - Synthesizer :	22
3 - Vocoder :	23
4 - Script python script bash	24
ENTRAINEMENT	25
1 - Encoder :	25
2 - Synthesizer:	25
3 - Vocoder :	26

PROBLÈMES RENCONTRÉS	27
SYNTHESE DES RESULTATS	29
1 - Encoder	29
2 - Synthesizer	30
3 - Résultats finaux	32
AXE D'AMELIORATION	33
CONCLUSION	34
1 - Apports individuels et collectifs	34
2 - Conclusion générale	34
BIBLIOGRAPHIE / WEBOGRAPHIE	35
ANNEXES	37
ANNEXE 1 : Méthode de construction de la base de donnée	37
ANNEXE 2 : Contrat d'accord d'utilisation de la voix dans le cadre des tests du projet M1	39
ANNEXE 3 : Les exigences des versions des logiciels	40

GLOSSAIRE

STT	Speech-To-Text (parole à texte).
TTS	Text-To-Speech (texte à parole).
SV2TTS	Speech Vector To Text-To-Speech.

Social Engineering L'ingénierie sociale est, dans le contexte de la sécurité de l'information, une pratique de manipulation psychologique à des fins d'escroquerie.

Framework Un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel.

EER Equal Error Rate, sert à prédéterminer la valeur limite pour le taux de faux positifs et le taux de faux négatifs. Lorsque ces deux taux sont égaux, la valeur commune correspond au taux de l'erreur égale (ERR). Plus cette valeur est faible, plus le système est précis.

INTRODUCTION

Qu'est ce le voice cloning ?

Le Voice Cloning est la création d'une simulation artificielle de la voix d'une personne. Cette parole synthétique ressemble étroitement à la voix humaine ciblée, la différence entre la vraie et la fausse voix est imperceptible pour la personne moyenne.

Cette technologie peut-être utilisée pour des trucages, canulars, mais aussi pour diffuser des fake-news. C'est pourquoi on parle de deep fake ou d'hyper trucage. Des personnes, mal intentionnées, ont déjà utilisé le deep fake audio pour arnaquer des personnes. Par exemple, en 2019 le directeur général d'une entreprise énergétique britannique s'est fait avoir par une arnaque de ce type. Il a envoyé 220.000 euros à un fournisseur hongrois après avoir reçu un appel téléphonique de son PDG. Celui qui aurait passé ce coup de téléphone aurait demandé un "transfert urgent et qu'il devrait être fait dans l'heure" (Les criminels n'ont toujours pas été identifiés)[1]. En effet, pour les hackers, de simples enregistrements audio suffisent à nourrir une intelligence artificielle. C'est une tendance qui est notable en cybersécurité ces dernières années. Les cibles des attaques ne sont pas que les logiciels ou les infrastructures. Mais c'est aussi les humains, c'est-à-dire les personnes qui travaillent sont la cible d'attaques (ça peut être des techniques de deep fake audio ou des attaques par phishing).

Cela pose un problème pour les personnalités publiques qui s'expriment régulièrement à la télévision ou sur Internet, de même que pour des appels téléphoniques de patrons concernant des résultats financiers ... etc. L'intelligence artificielle va pouvoir copier la voix de ces personnes. Plus les audios seront de bonne qualité, plus le résultat sera vraisemblable.

Mais on peut encore aller plus loin dans le trucage. On peut imaginer ajouter une vidéo pour accompagner cet audio. C'est comme ça quand 2019, BuzzFeed a pu voir faire la video sur le président Barack Obama qui insulte de "sombre merde" le président Donald Trump[2].



figure 1 : image du deepfake de Barack Obama

CAHIER DES CHARGES

Contexte

Dans le cadre des projets de M1, Le professeur Matthieu SAUMARD (l'encadrant de ce sujet), nous donne comme présentation générale : "Vous devrez réaliser un système qui permet de "cloner" une voix [...] ce système doit être basé sur l'intelligence artificielle [...] Vous entraînerez un tel système pour un discours en français".

Concrètement, nos objectifs pour ce projet sont premièrement, remplir tous les critères donnés par M SAUMARD, c'est-à-dire avoir quelque chose de fonctionnel pour le Français et présentable pour des journées porte ouverte de l'ISEN. Mais comme nous sommes deux étudiants qui suivent le domaine professionnel "cybersécurité", pour nous ce sujet représente une façon de développer nos connaissances en matière de cyber et d'appliquer ce que nous avons pu apprendre avec l'ISEN. Donc, dans un deuxième temps nous voulons tester notre projet dans des situations réelles. Nous sommes aussi conscients des contraintes légales du projet.

Quelques contraintes

En effet, les peines pour usurpation de l'identité d'un tiers est puni d'un an d'emprisonnement et de 15.000 euros d'amende selon l'article 226-4-1 du Code pénal [3]. Avec en vue de l'aspect numérique de notre format ça représente aussi un manquement à la loi informatique et libertés qui peut engendrer une peine jusqu'à 5 ans d'emprisonnement et 300.000 euros d'amende. Nous avons décidé que les voix qui vont être cloner via notre projet seront les voix des personnes qui utilisent l'outil et les voix des personnes qui ont donné un accord pour que leur voix soient utilisées dans nos tests. (vous pouvez retrouver l'accord en annexe)

Une autre contrainte, technologique cette fois-ci. Nous avons à notre disposition uniquement nos ressources personnelles. Nous y reviendrons dans ce rapport mais voici la configuration utilisée pour entraîner notre système est : "GTX 1660, ADM Ryzen 5 3600, 16Go RAM" à noter que ce genre de configuration n'est pas optimale pour faire l'intelligence artificielle mais suffisante si le temps d'entraînement du modèle est rallongé suffisamment.

Voici sur le schéma ci-contre, une comparaison des performances des différentes cartes graphiques sur le marché. La flèche bleue indique où se trouve la configuration utilisée pour le projet.

La dernière contrainte est le temps. En effet, le projet est constitué de 14 semaines dont 5 semaines qui était en parallèle des cours de domaine professionnel.

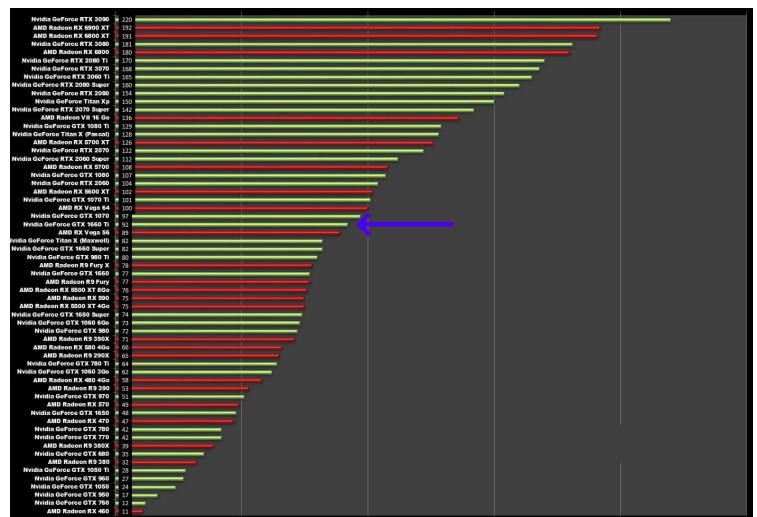


figure 2 : schéma comparaison des cartes graphiques

L'Analyse client

Les "clients", ou les utilisateurs qui sont susceptibles d'être intéressé par ce projet sont: premièrement les personnes en lien avec la cybersécurité et qui veulent faire de la sensibilisation. Ou les personnes qui veulent utiliser cet outil open source à des fins diverses. De plus dans le cadre des projets M1 et de la présentation que Monsieur SAUMARD a fait. Ce projet est susceptible d'être utilisé comme présentation des projets M1 lors de portes ouvertes de l'ISEN.

Les besoins de ces clients sont : Une interface graphique User-Friendly pour rendre facile l'utilisation. Et un code commenté et un README pour les personnes intéressées pour reprendre le projet open source. Le projet aura besoin aussi d'être performant dans le sens où il sera fonctionnel même avec des audios de courte durée.

Cas d'usage

Quelles sont les situations dans lesquelles, un projet comme celui-ci serait nécessaire ?

- Doublage post mortem d'acteur, de chanteur...
- Audit d'intrusion pour faire de la sensibilisation et informer les gens à ce genre de techniques Par exemple appeler des gens d'une équipe informatique pour soutirer des informations. Ou se faire passer pour un client et récupérer les informations du clients. Comme ce sont des attaques qui ciblent l'humain. L'une des défenses c'est la sensibilisation.
- Fins frauduleuses

Pour faire de la désinformation comme on a pu le voir dans l'exemple avec le président américain ou de l'usurpation d'identité dans le cadre de social engineering.

DESCRIPTION FONCTIONNELLE

1 - Partie Système

Ce projet va se baser sur une technique développée dans l'article “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”[4] rédigée par Ye Jia, Yu Zhang, Ron J. Weiss. Cette technique est appelée SV2TTS. c'est une manière de générer des TTS avec une voix choisie. Un TSS c'est un système informatique permettant de transformer un texte écrit en texte parlé. Différentes entreprises ont développé leur version du TTS et on peut remarquer que la plupart du temps, c'est une voix robotique. Tandis que le SV2TTS va reproduire, n'importe quelle voix. Le SV2TTS est composé des 3 parties entraînées individuellement que vous voyez ci-dessus. Et que cette partie va présenter :

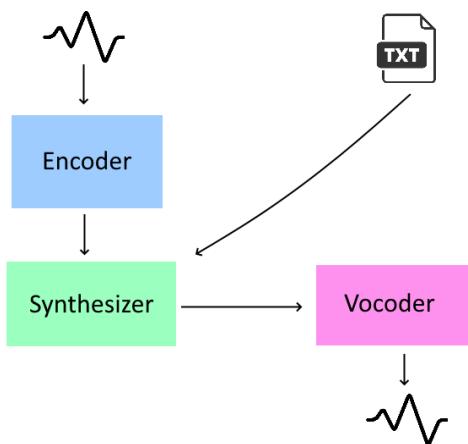


figure 3 : Schéma du fonctionnement du système

A - Encoder

Son travail va être de prendre en entrée la voix d'une personne (la voix à cloner à la fin). Et va faire une analyse et comprendre, comment la voix sonne et quelles sont ses caractéristiques si elle est grave ou aiguë, s'il y a un accent et avec quel ton la personne parle. Donc, ici, les mots qui sont prononcés par la personne ne sont pas intéressants. Pour représenter les signaux de la voix, le spectre des signaux ne sera pas sur une échelle hertzienne. En effet la différence entre 500 Hz et 1000 Hz est perceptible par tout le monde. Mais la différence entre 7500 Hz et 8000 Hz n'est quasiment pas perceptible à l'oreille. C'est pourquoi l'échelle Mel est intéressante à utiliser, qui est faite de façon à ce que tous les sons soient à la même distance les uns des autres. C'est le résultat d'une transformation non linéaire.

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right)$$

Voici ci-dessus la formule pour convertir des Hertz en Mels avec f la fréquence en Hz et m en mel.[5] Une fois avec cette représentation de la donnée, l'entraînement va être similaire à un réseau de neurones siamois. C'est-à-dire qu'il va comparer la donnée reçue avec une autre qu'il connaissait déjà pour savoir s'ils viennent tous les deux du même auteur. L'objectif est d'obtenir un vecteur d'informations de comment l'auteur sonne. Quelles sont les caractéristiques de sa voix.

B - Synthesizer

Son rôle est d'analyser l'entrée texte. Ce que la voix va dire. Pour pouvoir générer le spectrogramme mel que le vocodeur va ensuite convertir en son. Pour faire ça, il va mapper le texte entré avec des phonèmes (une unité qui permet de distinguer les mots les uns les autres).[6] Avec toutes ces données, le synthesizer va utiliser l'architecture Tacotron2 [7] pour générer un spectrogramme de la voix qui prononce les mots du texte d'entrée.

A la fin du synthesizer la voix qui est synthétisé sous forme de spectrogramme mel va être comparé avec celui qui est initialement dans l'encoder pour comparer et voir les pertes et les minimiser pour avoir un meilleur résultat.

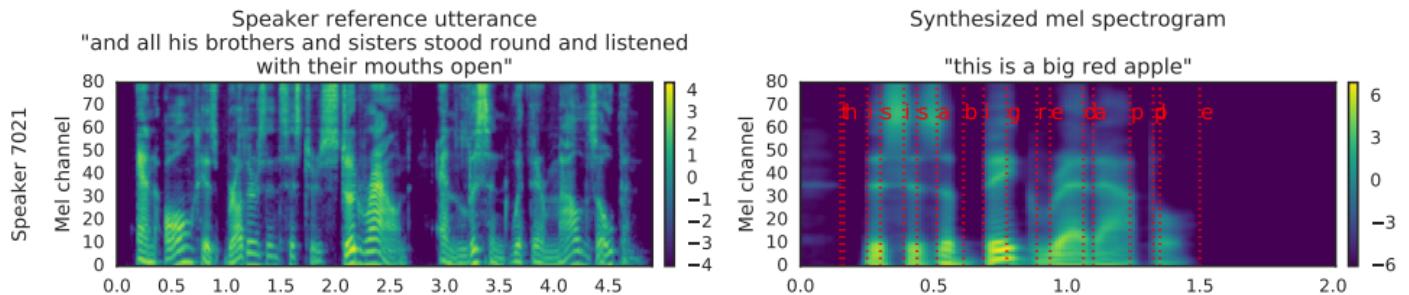


figure 4 : Illustration des mel spectrogramme

C - Vocoder

A partir du spectrogramme mel du synthesizer le vocoder va gener un audio. Il est basé sur le modèle WaveNet de DeepMind [8]. Composé de 30 couches de convolution dilatées. Le réseau n'est pas directement conditionné à la sortie de l'encodeur de haut-parleur. Le spectrogramme mel prédit par le synthétiseur

le réseau capture tous les détails pertinents nécessaires à une synthèse de haute qualité d'une variété de voix, permettant de construire un vocodeur multi-locuteurs en s'entraînant simplement sur les données de nombreux locuteurs.

2 - Partie Interface

Pour l'interface, Nous avons décidé d'utiliser le micro-framework Flask[9] (micro par sa taille). Car il est open-source et basé sur du python. Comme le système est lui aussi codé en python, la communication entre le système et l'interface est simplifiée. Et la communication entre l'interface et l'utilisateur va être une page web.

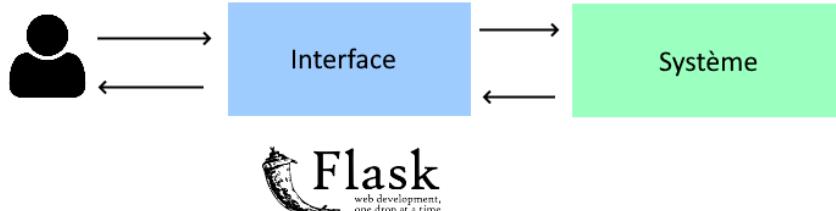


figure 5 : Schéma du fonctionnement de l'interface

L'interface est donc une page web local qui va faire la connexion entre l'utilisateur (celui qui va cloner des voix) et le système. Les options qu'on peut retrouver sur l'interface sont :

- Choix d'une voix à cloner
- par upload d'un fichier
- par enregistrement
- Choix du texte

Voici ci-dessous la maquette de l'interface avec les caractéristiques recherchées

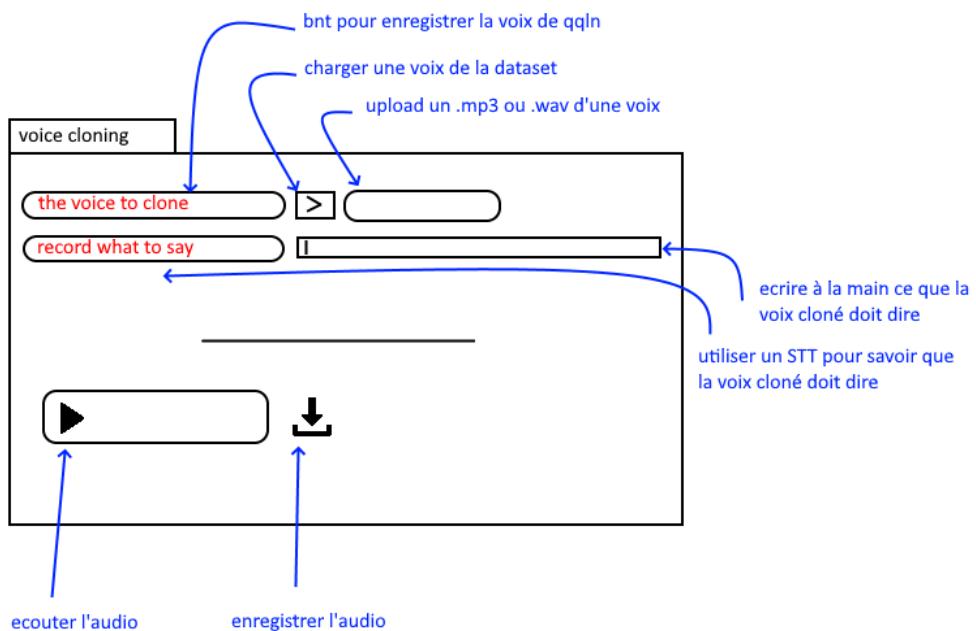


figure 6 : Maquette de l'interface

ANALYSE DU PROJET

Le projet de “Real-Time-Voice-Cloning” de Corentin Jemine, connu sous le pseudonyme de CorentinJ. Est un projet open source qui est une implémentation des l’articles :

- “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”.
- “Efficient Neural Audio Synthesis”
- “Tacotron: Towards End-to-End Speech Synthesis”
- “Generalized End-To-End Loss for Speaker Verification”

Ce projet est toujours d’actualité depuis octobre 2018, avec des mises à jour régulières et plus de 17 contributeurs. Ce projet fonctionne en temps réel. Les fichiers qu’on peut retrouver dans ce projet sont des fichiers pour entraîner les modèles de l’encodeur, du synthesizer et du vocoder. Il y a aussi de présent dans ce projet une interface graphique. Comme l’illustration ci-dessous le montre.

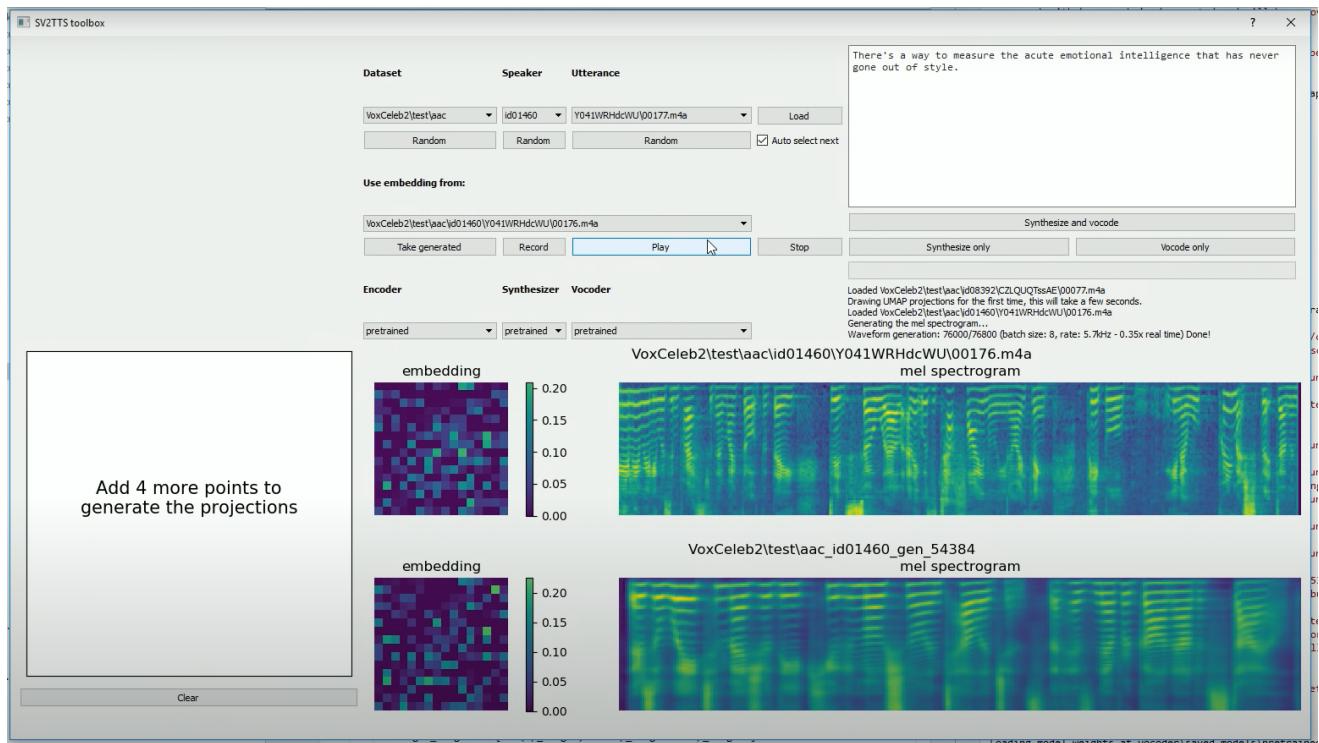


figure 7 : Capture d'écran du projet de CorentinJ

Les configurations du projets sont un système d’exploitation Windows et Linux, Un GPU est recommandé pour l’entraînement et pour la vitesse d’inférence, mais cela n’est pas obligatoire. Python3 est nécessaire (Python3.7, Python3.5 ou une version supérieure). FFmpeg, PyTorch (Dans sa dernière version la plus stable).

Le projet propose aussi des modèles qui sont pré-entraînés, sur les bases de données de LibriSpeech.

GESTION DU PROJET

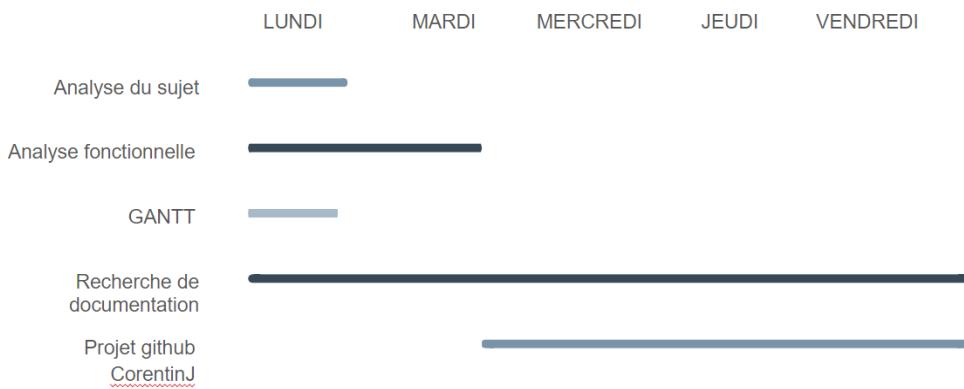
DÉROULEMENT DU PROJET

Voici comment nous nous sommes organisés pour la réalisation du projet. Tout d'abord nous allons présenter, dans quelles conditions s'est déroulé ce projet. Et Comment nous nous sommes répartis les tâches au sein du groupe. Ce projet s'est divisé en trois périodes, comme le montre le schéma ci-dessous



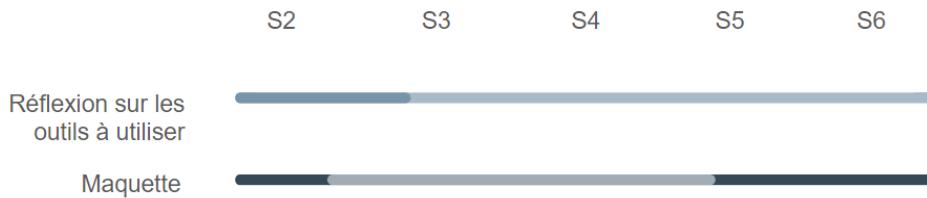
1 - Phase de Lancement

Dans cette phase de lancement pour la répartition des tâches, nous avons un peu tout partagé comme il s'agit du début. On avançait en même temps que les découvertes de chacun. On a posé les bases du projet et on s'est mis d'accord sur la façon de travailler. L'objectif de cette première partie était de s'approprier le sujet, comment cela fonctionne, trouver des projets qui existent déjà... etc. Monsieur Saumard nous avait partagé lors de sa présentation du sujet un projet GitHub qui faisait justement ce qu'on avait besoin l'un des objectifs était donc aussi de faire fonctionner ce projet sur nos machines.



2 - Domaine professionnel

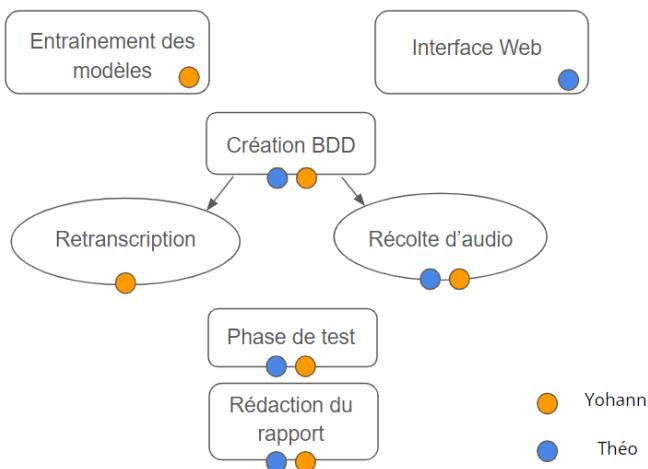
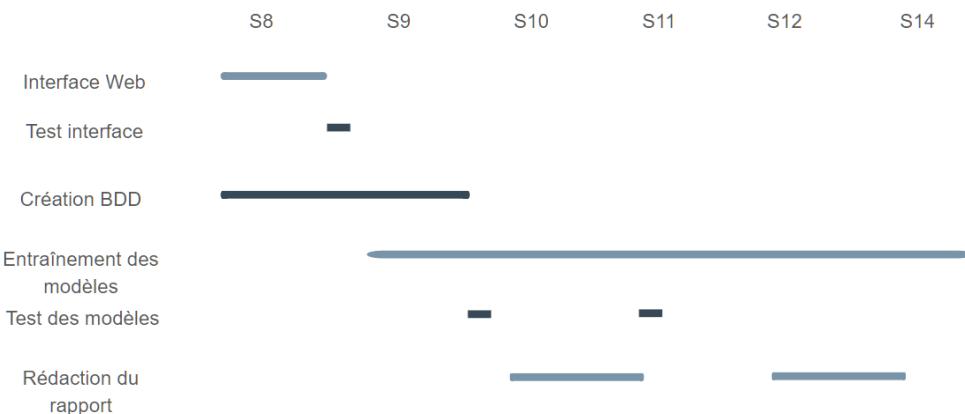
Une période moins productive pour le projet, comme nous étions occupés avec nos DP. Mais on a gardé le projet en veille. C'est à dire, on y pensait régulièrement, c'est comme ça qu'on a trouvé des idées de maquette et qu'on a testé des technologies pour savoir laquelle serait la plus intéressante pour le projet.



3 - Phase Finale

C'est dans cette période que c'est fait "le gros du projet". Nous devions nous occuper du développement de l'interface. De la création de la base de données. L'entraînement des modèles. Et faire nos tests. Avec nos contraintes, nous étions obligés de travailler à distance. Nous avons gardé le contact tout au long du projet, en partageant les avancées et les difficultés.

Voici comment nous nous sommes répartis les tâches. Et quand nous avons effectué ces tâches.



OUTILS DE VALIDATION TEST

Cette partie traite de la méthode de travail qui a été appliquée pour le développement du projet. Il existe deux méthodes de travail populaire pour ce genre de projet [x]: La méthode AGILE et la méthode en cycle V.

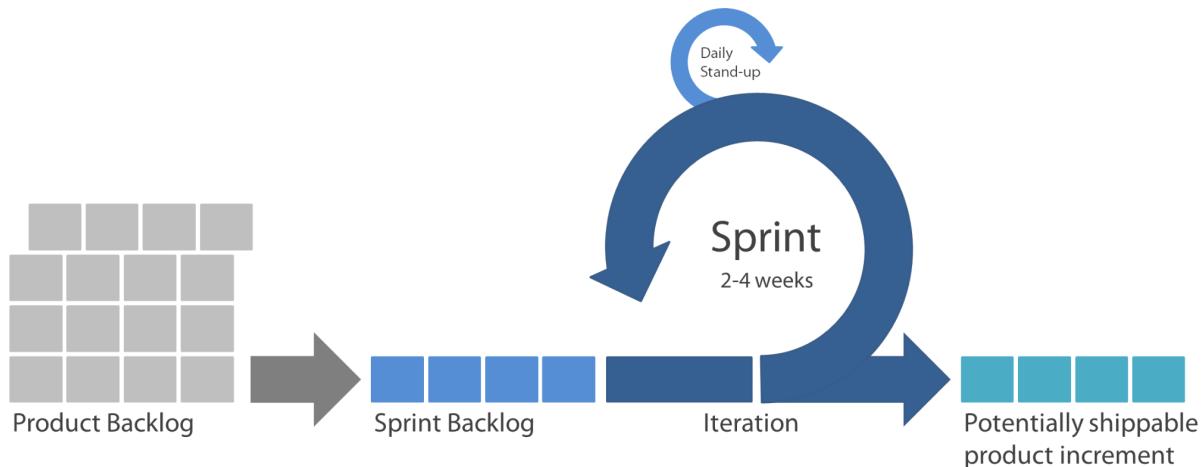


figure 8 : Schéma de la méthode AGILE

La méthode agile, va diviser le projet en plusieurs parties pour pouvoir avoir des objectifs à court terme. Comme le projet est divisé en plusieurs sous-projets. Une fois l'objectif atteint, on passe au suivant, et ce jusqu'à l'accomplissement de l'objectif final. Cette approche est plus flexible comme le montre l'illustration ci-dessus. Une autre méthode est celle des cycles en V. Il se caractérise par un flux d'activité descendant qui détaille le produit jusqu'à sa réalisation, et un flux ascendant, qui assemble le produit en vérifiant sa qualité. Ce modèle est issu du modèle en cascade dont il reprend l'approche séquentielle et linéaire de phases.

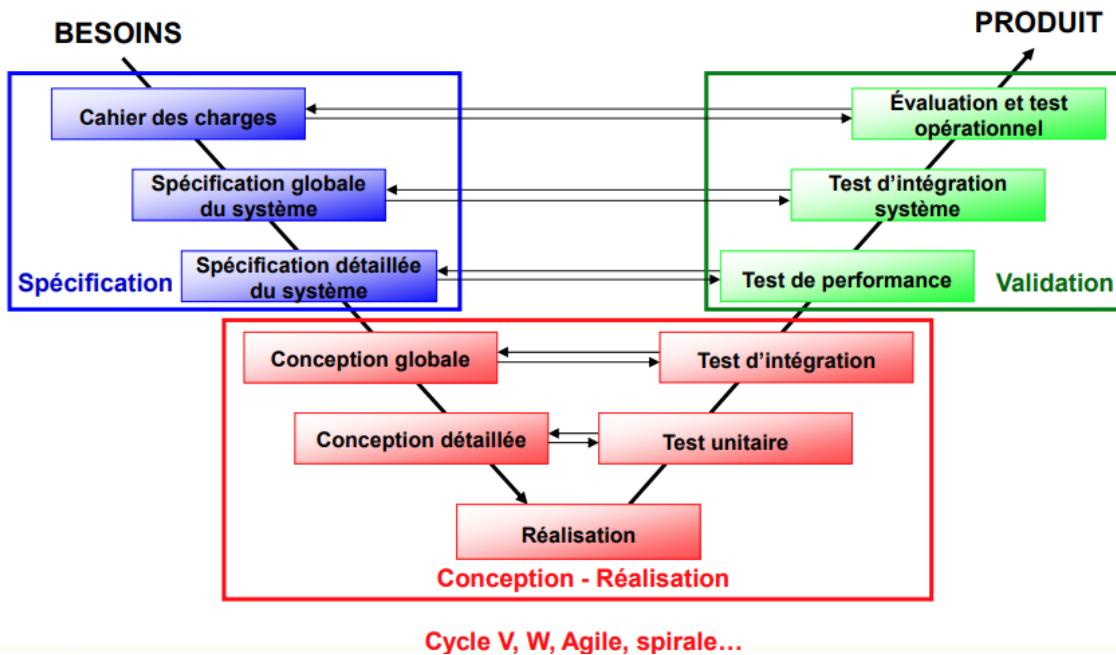


figure 9 : Schéma méthode en cycle V

Or, la méthode agile n'est pas réalisable dans le cadre de ce projet. Même si le projet Voice cloning était divisé en petite partie pour avoir des résultats à court terme tel que : Création de la Base de données, Entraînement des modèles, Création de l'interface, Mise en place dans des situation réelle. Comme toutes ces parties sont indépendantes les unes des autres, la méthode de cycle en V est plus cohérente.

COMPTE RENDU DE RÉUNION ET REPORTING

Cette partie présente le suivi des projets à travers les réunions qu'on a pu avoir avec le professeur référent du projet. Ces réunions se faisaient en distanciel sur la plateforme Google Meet. Les autres réunions entre les membres du groupe pour le projet se faisaient continuellement. Pour se tenir au courant des avancées de chacun et se poser des questions.

<u>Reunion # 1</u>	
Date	Mercredi 24 Novembre 2021
Contexte	Avant les affectations des sujets de projet
Sujet	Présentation du sujet et Questions
Synthèse	Présentation du sujet et des articles scientifiques sur le sujet. Présentation du projet de CorentinJ. Mise en place des objectifs du projet. Question : "Où va se trouver la difficulté principale de ce sujet" : Dans l'entraînement de la base de données.

<u>Reunion # 2</u>	
Date	Vendredi 7 Janvier 2022
Contexte	Fin de la phase de lancement
Sujet	Demande de matériel
Synthèse	Présentation des avancées du projet Demande d'utilisation de matériel du LabISEN pour entraîner nos modèles, qui sera refusé. Contre Proposition : Utilisation de Google Collab, qui n'est pas applicable pour notre base de données. Solution : Utiliser le matériel personnel pour entraîner les modèles. Même si cela implique d'entraîner les modèles plus longtemps. Cela peut poser un risque de manquer de temps à la fin du projet pour avoir des résultats.

Reunion # 3

Date	Jeudi 3 Mars 2022
Contexte	Phase Finale
Sujet	Demande de matériel : Logiciel de retranscription
Synthèse	Présentation des avancés du projet -Interface Utilisateur fonctionnel pour un discours en anglais. -Base de données incomplète. Présentation d'une solution pour finir de construire la base de données. Demande d'achat du logiciel.

Reunion # 4

Date	Mercredi 30 Mars
Contexte	Fin de Phase Finale
Sujet	Manque de résultat
Synthèse	Présentation des avancés du projet -Base de données fini -150h d'entraînement. Recherche d'une solution pour le manque de résultat. Solution proposé : Utiliser le modèle anglais déjà entraîné et entraîné par dessus avec notre bdd en français

DÉVELOPPEMENT TECHNIQUE

Après les parties de présentation du projet et des méthodes de travail. Cette a pour objectif de retranscrire comme s'est passé le développement du projet en détail. Pour laisser une trace des problèmes rencontrés, les solutions et les choix qui ont été faits.

Au début du projet, l'outil Google Collab a été choisi pour réaliser l'entraînement des 3 modèles et les différents tests du projet car c'est un outil intuitif pour exécuter du code Python directement sur le navigateur avec un accès GPU et un partage facile et peut être utilisé dans le cadre de l'intelligence artificielle.

Mais après quelques tests, un inconvénient est vite arrivé car l'upload de fichier sur Google Collab est très lent, par exemple un fichier de 500 MB a pris 30-40 minutes à s'uploader. Dans le cadre du projet, les deux bases de données réunies ont une taille de plus de 20 GB ce qui est conséquent et aurait demandé un temps énorme d'upload pour chaque test. La solution était d'upload les bases de données sur Google Drive et d'accéder à Google Drive avec Google Collab mais le problème de stockage se pose il faut assez de place pour les stocker et second problème le prétraitement des données et l'entraînement des modèles génère des fichiers supplémentaires. Ces fichiers constituent plus de 100 GB de données et il y a d'autant plus de ces fichiers qu'il y a de données dans les bases de données.

Un autre inconvénient était aussi présent, on a accès à du GPU et à de la RAM mais ceux-ci sont limités dans la version gratuite et bien que insuffisants pour le déroulement de notre projet. La version payante aurait pu être prise via un investissement pour le projet mais avec l'inconvénient cité précédemment, une autre solution a été choisie, celle d'utiliser un de nos PC contenant une carte GPU. La configuration de celui-ci est listée juste en dessous.

Configuration du PC :

Processeur : AMD Ryzen 5 3600 (6 coeurs CPU à fréquence de 3,6 GHz)

Carte Graphique : NVIDIA GeForce GTX 1660 (6GB de mémoire)

RAM : 16GB DDR4 3200MHz

A l'origine, pour développer l'interface, il y avait 2 technologies possibles et intéressantes. Flask la technologie qui a été retenue et PyQt. PyQt est un module libre qui permet de lier le langage Python avec la bibliothèque Qt. Il n'a pas été retenu bien qu'il pouvait répondre à toute les attentes de l'utilisateur. Mais là où Flask se différencie, c'est sur des outils de développement et de débogage.

Utilisation d'un logiciel de transcription STT : 360converterOfflineTranscriber. C'est un logiciel qui permet de transcrire une vidéo, un audio ou même une vidéo Youtube. On peut uploader les fichiers du PC directement sur l'application et il prend en charge différentes extensions sons et vidéos et également plusieurs langues notamment le français.

INTERFACE GRAPHIQUE

L'interface est composée d'une page principale que vous voyez ci-dessous. L'interface fonctionne à partir de deux framework :

- Flask, version 2.0.3 c'est un framework open source
- Bootstrap version 5.1.3 une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web

Projet M1 ReadMe About Us

Choix de la voix à cloner

Record a voice

StartStop

Upload a voice

Aucun fichier choisi

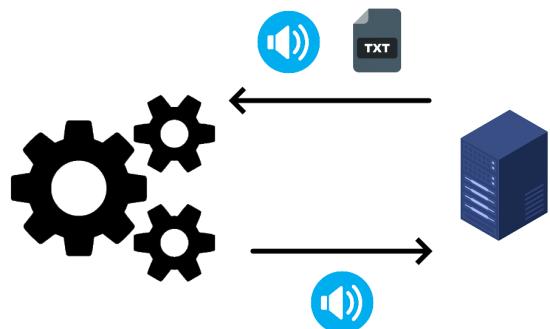
Que va dire cette voix ?

Envoyer au Système

figure 10 : Capture d'écran de l'interface du projet

Ces framework ont été choisi car ce sont des outils ou il existe une documentation importante avec une communauté qui est très développée. C'est-à-dire qu'il existe des exemples et des tutoriels de prise en main de ces outils qui facilitent le travail d'implémentation dans un projet comme celui-ci. De plus, c'est framework ont pu faire l'objet de cours dans la formation de l'ISEN.

L'objectif de l'interface est de faire la connexion entre l'utilisateur (celui qui va cloner des voix) et le système. comme le montre le schéma ci-contre. Quand l'utilisateur va envoyer les informations (audio et textuel) au système. l'interface sera dans l'attente d'une réponse (audio). Ce processus peut prendre quelques secondes.



1 - Fonctionnement Bootstrap

C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.[11]

Pour la partie CSS qui est à copier dans les balises <HEAD> de tous les fichiers html.

```
<link rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
      integrity="sha384-gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous">
```

Pour la partie JavaScript le code qui suit est conseillé d'être collé à la fin du code html, c'est à dire proche de la balise </BODY>

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
       integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
       crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
       integrity="sha384-U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01c1HTMGa3JDZwrnQq4sF86dIHNDz0W1"
       crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"
       integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM"
       crossorigin="anonymous"></script>
```

2 - Fonctionnement Flask

Pour que Flask fonctionne correctement. Il faut lui préciser le fichier qui va contenir la variable d'environnement “FLASK_APP” ici c'est le fichier python qui va contenir tout le fonctionnement de l'interface “__init__.py” dans le cas de ce projet. Il y a aussi d'autres variables d'environnement qui sont utilisées dans ce projet. FLASK_ENV en mode “development” pour avoir accès à tous les outils de développement. Et de même façon pour FLASK_DEBUG qui est assigné à la valeur “1” pour avoir accès à tous les outils de debug de Flask.

Pour lancer l'application il suffira de lancer les commandes suivantes ou exécuter le fichier __init__.py

\$ flask run	\$ python3 -m flask
--------------	---------------------

Flask va automatiquement ouvrir le port 5000 avec notre application

```
→ A4_PROJET_M1 git:(master) ✘ ./__init__.py
* Serving Flask app '__init__' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 351-151-158
```

figure 11 : Capture d'écran du terminal quand l'interface se lance

3 - Arborescence

Voici comment l'arborescence de l'interface est faite, Ce n'est pas un choix pour le projet c'est comment Flask est construit . Le fichier `__init__.py` qui a été présent dans la partie précédente doit se trouver dans le même répertoire que les répertoires suivants : `template` et `static`, qui vont tous deux contenir comme le schéma ci-dessous l'indique les fichiers html, javascript, css, textuel, image et video.

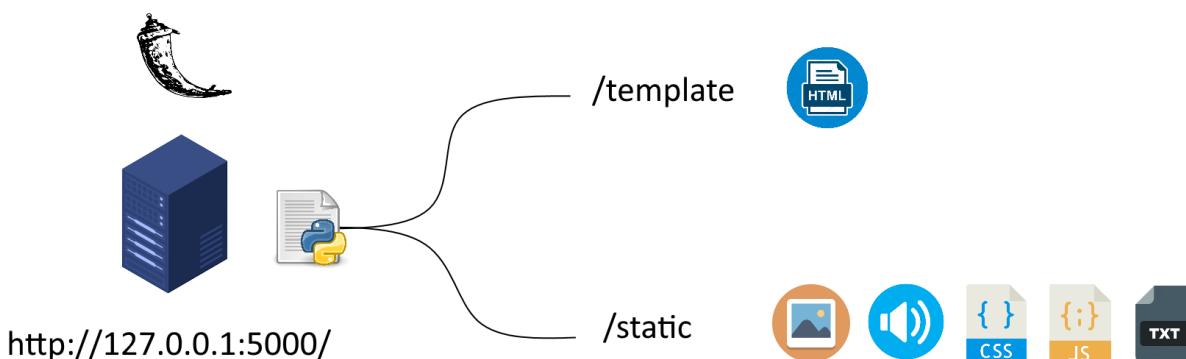
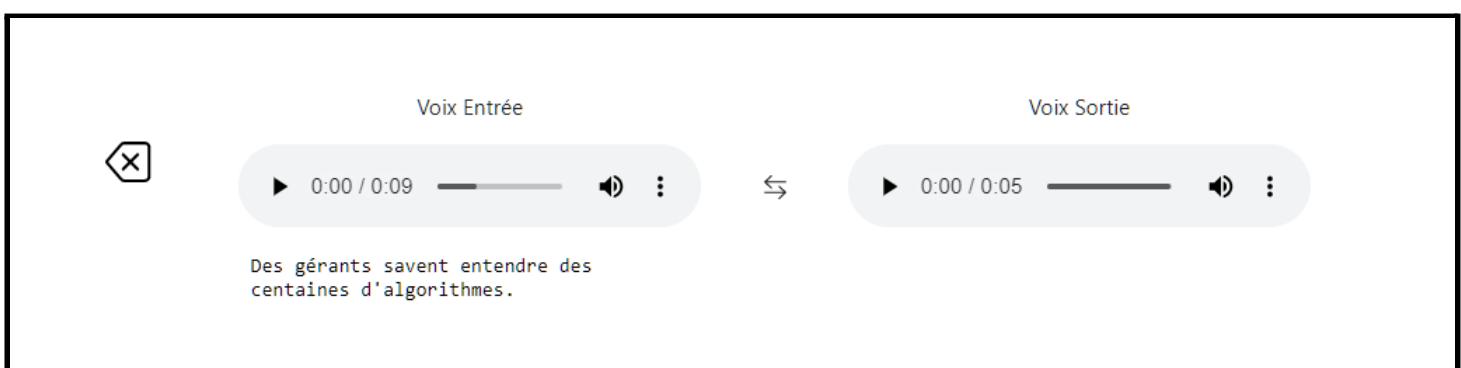


figure 12 : Illustration de l'arborescence de l'interface

Pour faire le lien d'une page à une autre et rendre le contenu des fichiers audio par exemple quand le système a terminer le traitement, l'interface utilise les fonctions suivante pour afficher la page des résultat :

```
render_template(results.html)    url_for("static", filename="output.wav")
```



4 - Caractéristique implémenté

A - Le choix de la voix

Par rapport à ce qui était prévu, le projet final aura 2 options pour choisir la voix qui sera cloner. Soit on va demander à l'utilisateur d'upload un fichier et qui sera stocker sous le nom de "audio_from_interface.wav" dans l'arborescence du projet. Sinon on demande à l'utilisateur d'enregistrer sa voix directement.

Le projet sera capable de cloner une voix à la fois.

Choix de la voix à cloner

Record a voice



Upload a voice

[Choisir un fichier](#) Aucun fichier choisi [Envoyer](#)

figure 13 : Capture d'écran de l'interface

B - Le Choix de l'audio

Ce projet sera capable d'enregistrer le texte que la voix cloner va prononcer. En écrivant dans l'espace "Que va dire cette voix ?" Cette donnée sera stockée sous forme de fichier texte dans le fichier "static/to_say.txt". L'option de la maquette "record what to say" n'a pas été développée finalement car elle n'apporterait pas grand chose de plus. Cela ajoutait de la difficulté supplémentaire car pour faire un STT comme celui là. Le projet aurait eu besoin d'un autre modèle à entraîné.

C - Autre Informations

Une fois que ces informations sont remplies. Le traitement du système est possible. Et après quelque seconde, les résultats s'affichent sur une autre page. De plus, sur l'interface, il est possible de retrouver des informations sur le groupe ou sur le projet en général.

BASE DE DONNÉES

1 - Encoder :

Pour l'encoder, il fallait une base de données avec beaucoup de voix différentes sans avoir besoin d'une transcription de ce qui est dit. Sur Internet il y a la base de données common-voice de Mozilla [10] qui regroupe plus de 15000 voix et représente 826 heures d'audio. La base de données était stockée comme montré sur le schéma avec un fichier par personne contenant tous les audios de la personne permettant au modèle durant l'entraînement de différencier les différentes voix.

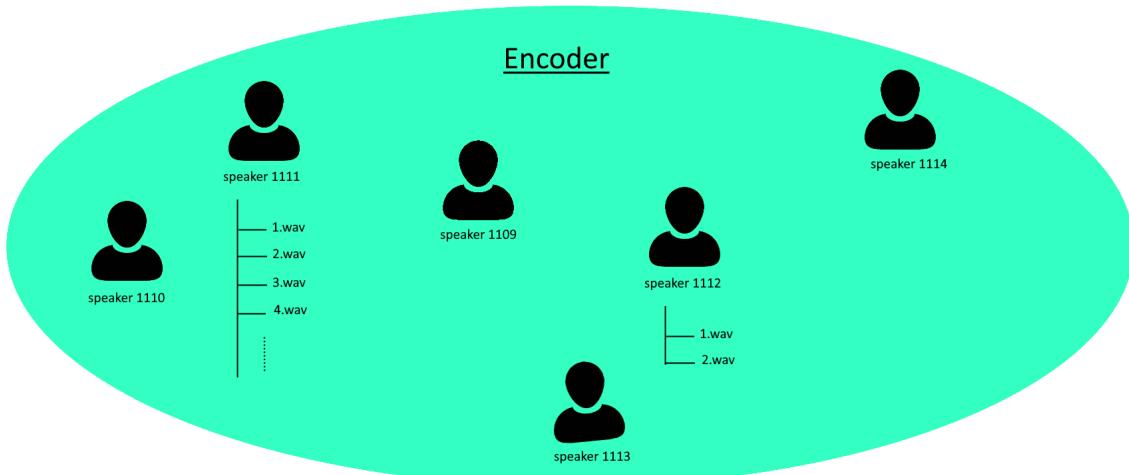


figure 14 : Schéma de l'arborescence de la base de données encoder

2 - Synthesizer :

Pour le synthesizer, il fallait également une base de données avec des voix différentes et en plus du fichier audio, il fallait un fichier texte avec la transcription de l'audio. Pour faire cette base de données, il a été utilisé des petites bases de données trouvées sur Internet tel que "openslr" contenant quelques heures d'audio avec la transcription mais même avec une trentaine d'heures en tout ce n'était pas suffisant pour avoir des résultats satisfaisants. Pour remédier à ce problème, il a fallu récupérer de nombreux extraits de livres audio du site Audible, avec environ 45 minutes d'audio récoltées par lecteur s'il y avait un nombre suffisant d'extraits. Ensuite tous les extraits d'un même lecteur ont été rassemblés en un long audio pour avoir à transcrire qu'un audio par lecteur au lieu de plein de petits extraits ce qui est un gain de temps énorme. Du coup pour la faire la transcription le logiciel STT "360converter" a été utilisé . Pour pouvoir utiliser le logiciel sans limite de temps et de taille sur les audios, la licence du logiciel a dû être achetée et grâce à ce logiciel tous les audios ont pu être transcrits. La transcription a été copié dans un fichier texte avec les timecodes et le texte associé et avec un script python on a créé un fichier texte pour chaque timecode avec le texte associé et un fichier texte contenant tous les timecodes pour les utiliser comme marqueurs dans le logiciel Audacity et ainsi couper l'audio en fonction de ces timecodes.

00:00:00 - 00:00:20
comme il plaira à monsieur trois secondes avant l'arrivée de la lettre de j b obscene je ne songeais plus à poursuivre la licorne qu'à tenter le passage du nord-ouest trois secondes après avoir lu la lettre de l'honorable secrétaire de la marine je comprenais enfin que ma véritable vocation l'unique but de ma vie était de chasser ce monstre inquiétant et dans pure

00:00:20 - 00:00:40
le monde cependant je revenais d'un pénible voyage fatiguées avide de repos je n'aspirais plus qu'à revoir mon pays mes amis mon petit logement du jardin des plantes mes chères et précieuses collections mais rien ne put me retenir j'oubliais tout fatigue hamid collections et j'ai accepté sans plus de

Il y a eu quelques problèmes sur cette base de données avant de trouver cette solution, tout d'abord il avait été décidé de récupérer des livres audios connus permettant de récupérer la transcription du livre directement sur Internet. Le problème est que les audios avaient une durée de plus 10 minutes et qui pouvaient aller jusqu'à 1h30 ce qui faisait des fichiers beaucoup trop lourds pour l'entraînement du synthesizer. La solution était de découper l'audio mais le faire à la main aurait pris trop de temps car il aurait fallu découper phrase par phrase. Il a fallu chercher des logiciels STT pour transcrire les audios automatiquement mais il fallait également les timecodes pour découper l'audio facilement, une solution avait été trouvé avec une nouvelle fonctionnalité de Word transcribe qui permettait de transcrire un fichier audio et il donnait les timecodes d'une durée 20 secondes environ et le texte correspondant en dessous. Mais l'inconvénient étant que la fonctionnalité transcribe est limitée à 300 minutes par compte ce qui était insuffisant pour réaliser la base de données. C'est pour cela que le logiciel 360converter a été choisi et utilisé par la suite.

L'arrangement des fichiers dans la base de données est le même que pour la base de données de l'encoder, il y a un fichier par personne et chaque fichier contient les audios de cette personne avec les fichiers textes contenant la transcription des audios.

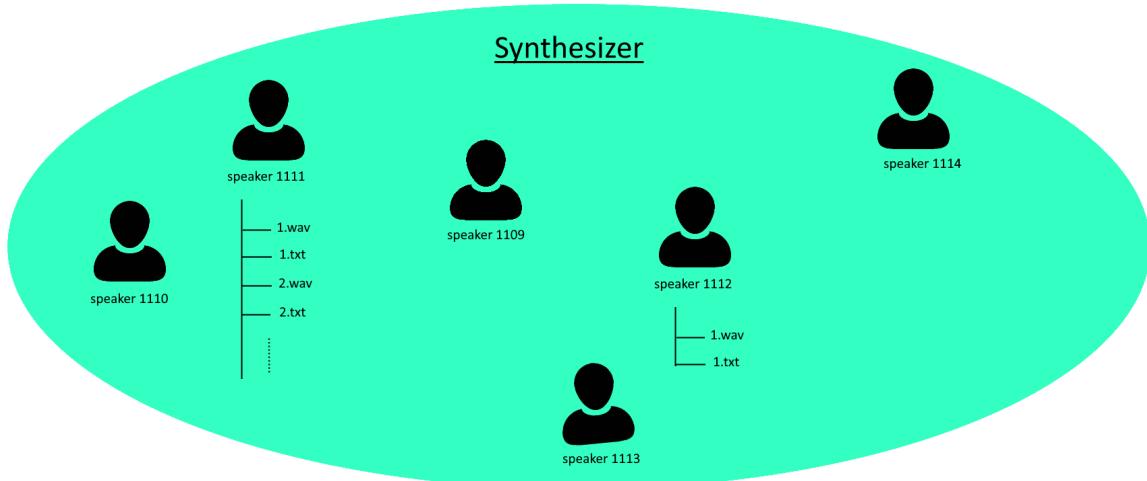


figure 15 : Schéma de l'arborescence de la base de données synthesizer

3 - Vocoder :

Le vocoder n'a pas besoin d'une base de données à proprement parler car il va utiliser les données générées par le synthesizer lors de son entraînement. Le vocoder a été entraîné à partir des données du synthesizer au début mais à la fin il a été choisi d'utiliser directement le vocoder entraîné sur le github permettant un gain de temps en évitant de devoir l'entraîner.

4 - Script python script bash

Pour avoir un gain de temps sur la création de la bdd, il a fallu créer des scripts python et bash car avec la quantité de données traitée faire tout à la main aurait demandé beaucoup trop de temps déjà que la récupération d'audio pour la base de donnée était longue. Ils ont été utiles pour des tâches simples comme créer les différents dossiers dans la base de données comme pour des tâches plus complexes.

Après avoir rassemblé tous les audios en un seul pour chaque lecteur et réalisé la transcription de ces audios, il fallait découper le texte avec la transcription en plein de petits fichiers textes et également récupérer les timecodes des différents textes pour pouvoir ensuite découper l'audio sur Audacity en important ces timecodes. Pour faire cela, un script python a été réalisé qui parcourt chaque fichier de la base de données et qui lit le fichier texte transcrit ligne par ligne. Une ligne sur deux il écrit dans un autre fichier texte le timecode convertit en seconde pour pouvoir être lu par Audacity et la deuxième ligne sur deux il crée un fichier texte avec le texte lu dans le fichier transcrit.

```
file = open('transcript.txt', 'r')
file2 = open('marqueur.txt', 'w')
line = file.readlines()

for i in range(0,len(line),2):      #lit les timecodes et les convertit en seconde
    tmp = line[i][:len(line[i])-1]  #pour réaliser des marqueurs compatible avec Audacity
    time = tmp.split(" ")[0].split(":")
    minute = int(time[1])
    seconde = int(minute)*60 + int(time[2])
    file2.write(str(seconde)+"\t"+str(seconde)+"\n")

for i in range(1,len(line),2):      #crée un fichier texte pour chaque timecode
    tmp = line[i][:len(line[i])-1]
    nb = int((i-1)/2)+1
    if nb<10:
        filetmp = open('out-0'+str(nb)+'.txt', 'w')
    else:
        filetmp = open('out-'+str(nb)+'.txt', 'w')
    filetmp.write(tmp)
    filetmp.close()

file.close()
file2.close()
```

marqueurs.txt - Bloc-notes	
Fichier	Édition
0	0
20	20
42	42
60	60
80	80
97	97
123	123
142	142
162	162
178	178
193	193
208	208
228	228
248	248
263	263
291	291
312	312
332	332
352	352
369	369
388	388
408	408
440	440
458	458
475	475
495	495
515	515
534	534
557	557
577	577
599	599

figure 16 : Capture d'écran d'un script python et du fichier de marqueurs résultant du script

Un script bash avait également été utile au début de la base de données lors de la récupération de textes de livres connus sur des sites tels wikisource. Il y avait dans les textes des caractères spéciaux, des sauts de ligne ou encore des notes de bas de page annotées dans le texte et ces différents caractères étaient gênant pour le bon entraînement du modèle. Pour éviter cette situation, il a fallu créer un script bash pour supprimer ou remplacer ces caractères dans le texte original.

ENTRAINEMENT

Pour chaque module à entraîner, encoder, synthesizer et vocoder, il faut faire un prétraitement des données pour qu'elles soient lisibles et utilisables par l'intelligence artificielle. Cela rajoutait un temps supplémentaire à l'entraînement car à chaque fois qu'on modifiait la base de données de l'encoder ou du synthesizer il fallait refaire le prétraitement des données.

Après ce prétraitement, l'entraînement du modèle peut être lancé et les sorties générées par l'entraînement du modèle permettent de surveiller son avancement jusqu'à obtenir des résultats satisfaisants.

Tout le déroulé des commandes et des étapes pour entraîner le modèle de zéro est expliqué sur le github du projet initial dans la partie "Preprocessing and training".[14]

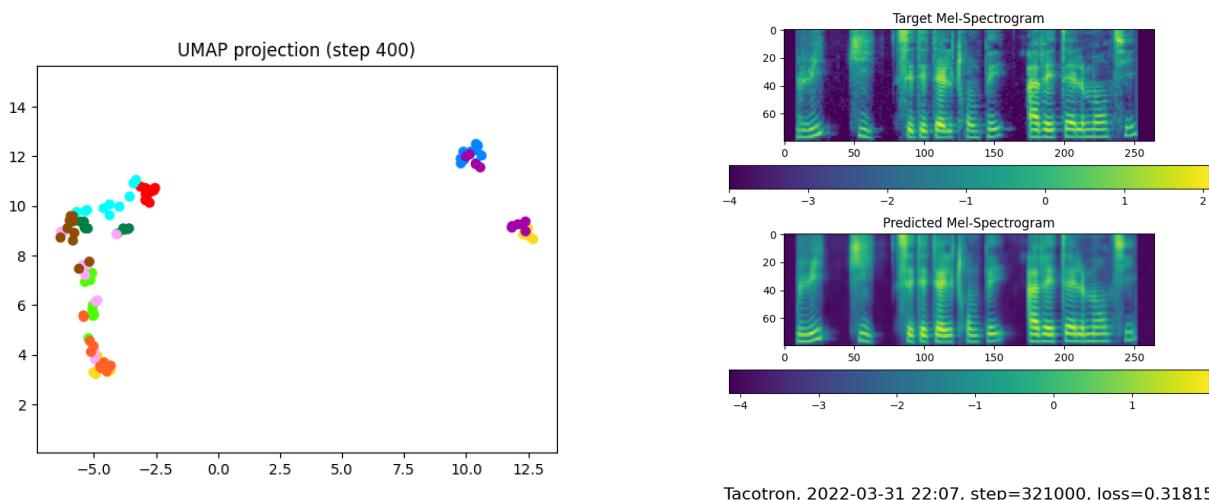


Figure 17 : à gauche image d'une projection UMAP de l'encoder et à droite image d'un mel-spectrogram prédict par le synthesizer

1 - Encoder :

L'encoder n'a pas eu besoin de beaucoup d'entraînement seulement quelques fois on suffit car la base de données de celui-ci était déjà bien conséquente et suffisamment grande pour avoir des résultats assez rapidement sans avoir besoin de rajouter des données dans la base et du coup d'éviter le prétraitement des données à chaque nouvel entraînement. Le temps d'entraînement a été d'environ 50h pour ce module.

2 - Synthesizer:

C'est sur cette partie-là qui a causé le plus de difficultés et qui a pris énormément de temps. Plusieurs tests ont dû être effectués pour l'entraînement du synthesizer par exemple avec une base de données réduites ou avec une base de données avec des audios longs. Pour la base de données finale, deux tests ont été réalisés, un premier à partir d'un modèle vide et un second à partir du modèle entraîné en anglais.

Pour le premier test avec la base de données réduites, le prétraitement et l'entraînement des données ont été rapides du fait du peu de données, ça a pris une vingtaine d'heures.

Ensuite un second test a été fait avec des audios longs, le prétraitement a pris environ 5h sauf que l'entraînement n'a pas pu être fait car les audios étant trop volumineux, cela demandait une puissance de calcul trop importante.

Un autre test a été effectué en ajoutant quelques audios dans la base de données grâce à la fonctionnalité transcribe de Word. La durée du prétraitement était la même que le test précédent avec 5h et l'entraînement a duré 72h pour ce test.

Pour l'avant-dernier test, la base de données utilisée est la base de données finale qui regroupe la base de données précédentes avec en plus 160h d'audio audible transcrits à partir du logiciel 360converter. Le prétraitement a été plus long du fait du plus grand nombre de données à traiter, il a duré 7h. L'entraînement a duré à peu près 150h, l'entraînement a été beaucoup plus long pour avoir les meilleurs résultats.

Enfin le dernier test, la même base de données que précédemment a été utilisée mais au lieu que l'entraînement commence d'un modèle vide, cette fois l'entraînement se base sur le modèle entraîné en angalis présent sur le github ainsi les poids du modèle entraîné sont conservés et le modèle doit juste apprendre les sonorités de la langue française. La base de données étant inchangée, il n'y a pas eu de prétraitement des données à faire et l'entraînement va être aussi long que le test précédent d'environ 150h.

3 - Vocoder :

Pour l'entraînement du vocoder, les données utilisées sont les données générées en sortie du synthesizer mais il est quand même nécessaire de réaliser le prétraitement de ces données. Comme des tests avec différentes bases de données ont été réalisés avec le synthesizer, le vocoder aussi a dû être entraîné plusieurs fois, 3 fois pour être précis.

Pour les 3 tests, le prétraitement des données a été réalisé en une durée de 4 à 5h et pour l'entraînement, comme avec le synthesizer, on a augmenté la durée de l'entraînement au fil des tests. La durée d'entraînement a varié de 15h pour le premier, à 30h pour le deuxième jusqu'à 72h pour le dernier.

Après réflexion et avec le peu de temps qu'il restait, il a été décidé d'arrêter d'entraîner le vocoder et d'utiliser directement le vocoder entraîné présent sur le github. Cette décision a permis un gain de temps pour se focaliser sur l'entraînement du synthesizer, une grosse partie du projet et aussi on sait que le vocoder est fonctionnel et que le vocoder n'est pas affecté par la langue car son rôle est de générer un audio à partir d'un mel-spectrogram.

PROBLÈMES RENCONTRÉS

Différentes difficultés ont été rencontrées durant ce projet, la première a été la durée du projet car le temps est passé très vite. Notamment la création et la finalisation de la base de données nous a pris 2-3 semaines et l'entraînement des trois modèles et le prétraitement des données a vite fait de remplir le temps qu'il restait. Cela nous amène à la seconde difficulté, la longueur de l'entraînement, dû à plusieurs facteurs surtout la configuration matérielle, avant d'avoir un résultat. Cela a eu pour conséquence une marge de manœuvre très courte car le temps de comprendre ce qu'il nous manquait et d'y remédier, beaucoup de temps était déjà passé sur l'entraînement.

D'autres problèmes ont été rencontrés sur la base de données du synthesizer, tout d'abord la création de celle-ci il a fallu créer notre base de données car les bases de données présentes sur Internet étaient insuffisantes pour un entraînement satisfaisant du synthesizer. Cela amène d'autres problèmes car pour le synthesizer il nous faut des audios avec le texte transcrit. Or, récupérer des audios à droite à gauche et transcrire le texte à la main ça demande un temps faramineux qui n'était pas disponible. La première idée a été de récupérer des textes de livres connus et leur équivalent audio sauf que ces audios étaient beaucoup trop longs pour le synthesizer et demandait une puissance de calcul supérieure à la puissance disponible. Et comme expliqué dans la partie base de données on a d'abord utilisé la fonctionnalité transcribe de Word puis on a trouvé le logiciel 360converter pour réaliser la transcription de nos audios.

Mais ce logiciel a un petit inconvénient dans l'utilisation faite pour ce projet, c'est qu'il réalise des timecodes d'une durée de 20 à 30 secondes voire un peu plus or dans le projet récupérer il y a une limite maximum de la taille du mel spectrogram généré à partir d'un audio. Cette limite équivalait à environ 14 secondes d'audio donc il a fallu augmenter cette limite de plus de deux fois sa valeur d'origine. Ce qui a causé une durée plus longue de prétraitement des données car il y a plus de données et des données plus volumineuses et également un ralentissement de la vitesse d'apprentissage du modèle ainsi un entraînement plus long. Avant la vitesse d'apprentissage était de 0,5 étape/s et la vitesse descendait à 0,28 étape/s soit presque un temps deux fois plus long pour un même nombre d'étapes d'apprentissage.

Le dernier entraînement a été l'entraînement à partir du modèle anglais déjà entraîné sauf que dans la langue anglaise il n'y a pas d'accent donc le synthesizer a été configuré avec une liste de caractères sans aucun caractères accentués sauf que dans les différentes transcriptions d'audio on retrouve des caractères accentués. Les caractères accentués sont supprimés lors de l'apprentissage car ils ne font pas partie de la liste de caractères et même en ajoutant les caractères accentués dans cette liste, cela ne fonctionne pas car la taille de la liste rentrée est différente de la taille de la liste avec laquelle le modèle s'est entraîné donc le programme s'arrête. Cela nous limite donc à ne pas pouvoir utiliser de caractères accentués avec ce modèle.

Sur la partie de l'interface, l'une des difficultés qui a été rencontré dans ce projet a été. C'est la différence entre le développement d'une page web "normale" et avec un framework python pour faire du développement web. En effet, les liens entre les pages ne se font pas de la même manière. Ou l'affichage du contenu d'une image sur la page. "La source" l'endroit où l'image va être cherché dans l'arborescence du projet pour être ensuite affiché fonctionne lui aussi pas de la même façon. Il faut définir des variables en utilisant Jinja2. Il est important de savoir que les accolades doubles extérieures ne font pas partie de la variable, mais de l'instruction print. On va donc indiquer la source de l'image avec une variable qui appelle la fonction flask.url_for, qui prend en paramètre le nom d'une vue (le nom de la fonction, dans une chaîne de caractères), avec ses éventuels paramètres, et retourne l'URL correspondante.

Affichage d'une image en html	Affichage d'une image avec flask

De même manière la transition entre les pages :

Navigation de page en page en html	Navigation de page en page avec Flask
	render_template("temp.html") ou

De plus pour l'interface, l'une des améliorations qui était prévu mais qui a posé problème c'était la page de chargement. Cette page intervenait après les entrées de l'utilisateur pour la voix et le texte qui est à cloner, pendant que le système était en train de traiter ces informations et l'affichage des résultats. Le souci c'est qu'avec Flask, il n'est pas possible d'appeler le programme qui va lancer le traitement et l'affichage de cette page en même temps.

Comme ce temps n'est pas très important, il est de l'ordre de quelque seconde. Si l'utilisateur a conscience de ce petit délai. La page de chargement n'est pas indispensable.

SYNTHESE DES RESULTATS

Cette partie va être découpée en 3 parties, la première partie va concerner les résultats sur l'encoder, la seconde les résultats du synthesizer et la dernière va être les résultats finaux obtenus lors du fonctionnement du projet.

1 - Encoder

L'encoder comme dit précédemment n'a pas eu besoin de beaucoup d'entraînement pour obtenir des résultats satisfaisants. Pour suivre ses résultats, il y avait la possibilité de lancer un serveur visdom sur le PC et ensuite sur l'adresse IP localhost d'avoir accès à différents graphes par exemple un graphique du critère d'évaluation EER et un graphique de la perte. Les graphiques sont mis à jour toutes les 100 étapes.

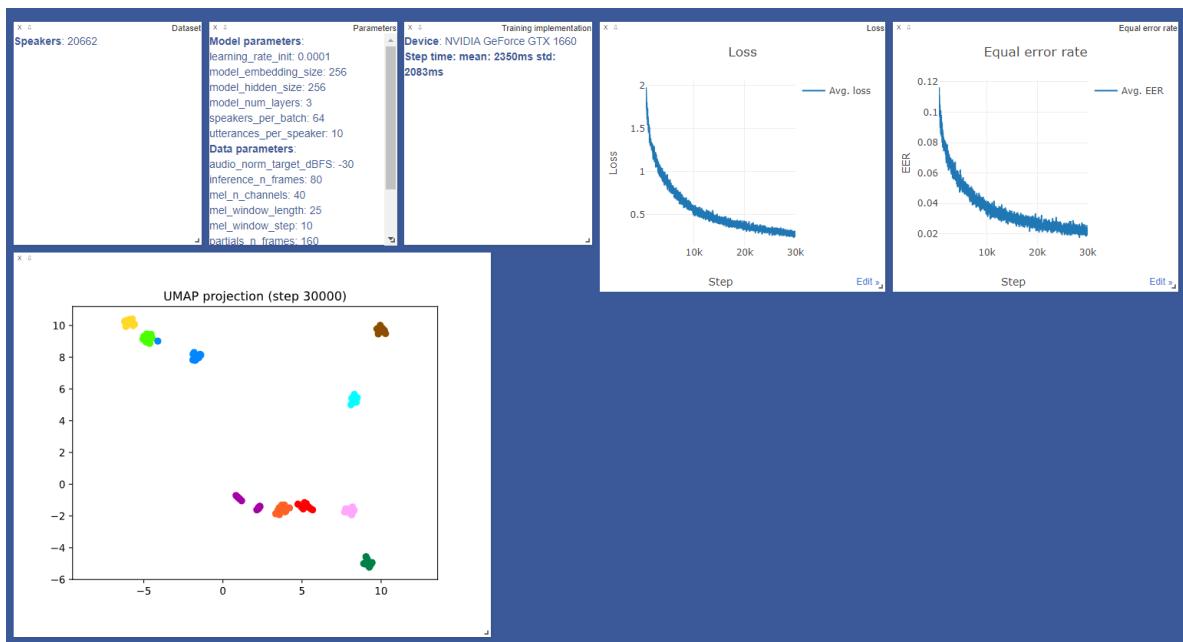


Figure 18 : image de l'interface des résultats présente avec le serveur visdom

Les deux courbes mentionnées précédemment, EER et perte, sont décroissantes. Elles montrent que le modèle s'améliore au fur et à mesure de l'entraînement. Ce qui est le plus intéressant et le plus parlant pour les résultats est le graphique en bas à droite, projection UMAP, avec les points de couleurs. L'IA va prendre les audios de 10 personnes, représentés chacun par une couleur, et va placer les audios (un point) sur le graphique en fonction des caractéristiques de la voix sur l'audio. Un résultat satisfaisant est donc quand tous les points de même couleur sont rassemblés au même endroit.

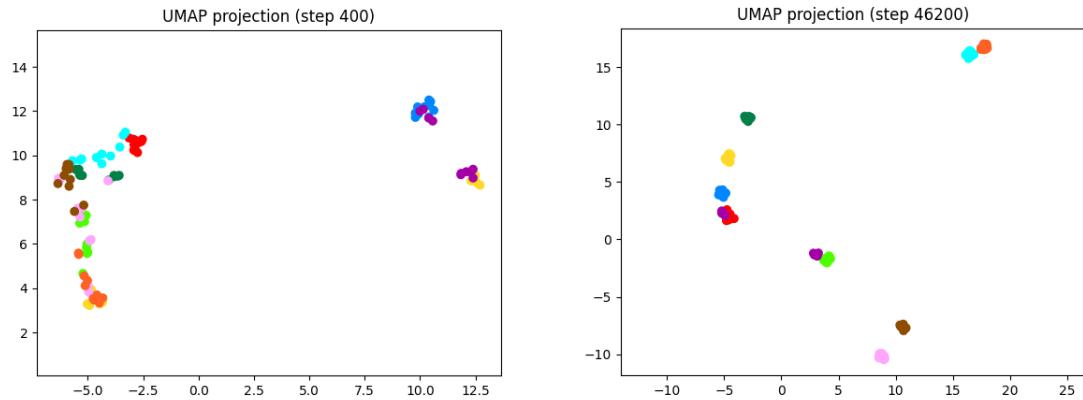
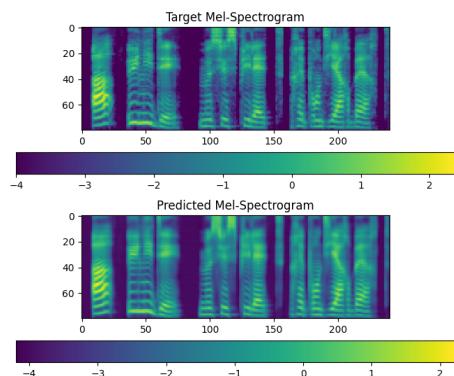


Figure 19 : projection UMAP à différentes étapes de l'entraînement de l'encoder

Sur le graphique de gauche, l'entraînement est à l'étape 400, qui correspond au début de l'entraînement, et les points de même couleurs sont éparpillés et se mélangent avec les autres couleurs même s'ils se trouvent dans des zones assez proches, ce n'est pas suffisant pour être un bon résultat. Alors que sur le graphique de droite à l'étape 46200, correspondant à la fin de l'entraînement, les points de même couleur sont tous rassemblés dans une petite zone et les différentes voix (couleurs) sont distinguables facilement. L'entraînement a été arrêté après cette étape car les résultats obtenus étaient ceux attendus pour le modèle.

2 - Synthesizer

Les résultats sur la partie synthesizer sont très mitigés car les résultats sont plus durs à analyser pour savoir si on a un bon résultat. L'entraînement du synthesizer génère un diagramme d'attention et une image du mel spectrogram généré et du mel spectrogram original avec la valeur de perte afficher. L'image des deux mels spectrogram ne nous aident pas à savoir si les résultats sont bons car il va juste nous montrer la prédiction d'un mel spectrogram et même si la perte est faible ça ne garantit pas un bon fonctionnement.



Tacotron, 2022-03-30 14:48, step=210000, loss=0.32563

Figure 20 : image d'un mel-spectrogram et du mel-spectrogram prédict par le synthesizer

Le facteur le plus important est le diagramme d'attention, car il nous permet de savoir le bon apprentissage du synthesizer. Pour comprendre comment fonctionne le diagramme d'attention, il faut reprendre le schéma du synthesizer et s'intéresser plus particulièrement à la partie encoder et décoder.

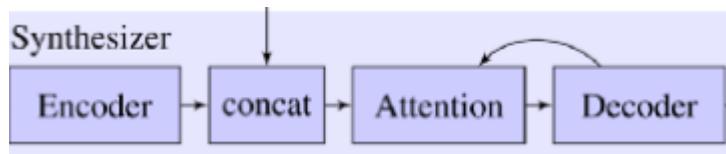


Figure 21 : Schéma du synthesizer

L'encoder prend le texte en entrée et lit un caractère et son état actuel à chaque étape et va générer en sortie un vecteur. Ce vecteur va servir de donnée d'entrée pour le decoder et il va générer une trame audio (mel-spectrogram). À chaque étape, il va décider quels vecteurs sont importants à la création de la trame audio.

Maintenant le diagramme d'attention, les ordonnées correspondent aux vecteurs générés par l'encoder et le nombre de vecteurs va dépendre de la longueur de la donnée d'entrée. Les abscisses représentent une étape du decoder lors de la création de l'audio frame. Les couleurs représentent l'utilisation des vecteurs à chaque étape.[15] Pour faire simple sur une étape, la somme des ordonnées vaut 1, plus un vecteur sera utilisé plus il sera de couleur jaune à l'inverse il sera bleu foncé/noir.

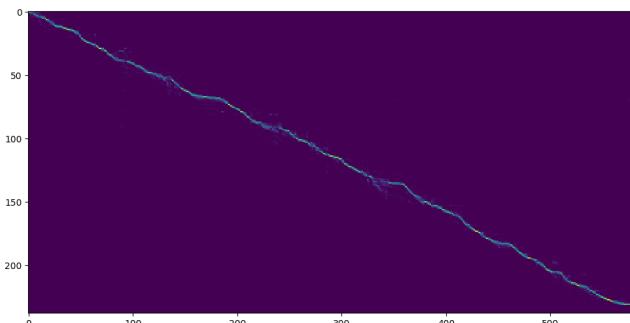


Figure 22 : image du diagramme d'attention attendu

Un bon alignement diagonal, signifie qu'un son "A" généré par le decoder est le résultat du choix du vecteur généré par l'encoder lors de la lecture du caractère "A". La ligne diagonale est le résultat quand l'audio est créé à partir du bon caractère d'entrée.

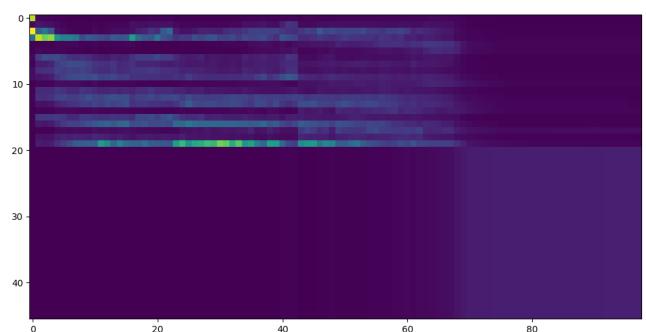
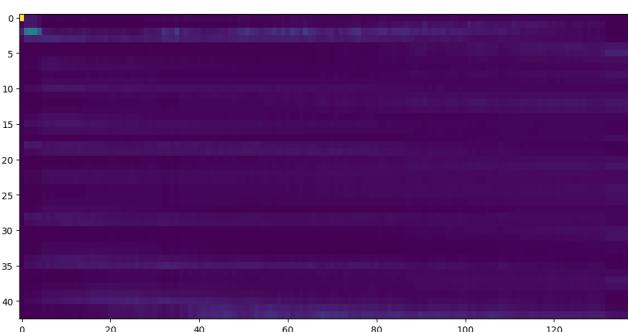


Figure 23 : image du diagramme d'attention à différentes étapes de l'entraînement du synthesizer
(à gauche étape 8000 et à droite étape 143000)

Les résultats obtenus avec tous les tests effectués avec les différentes bases de données ne sont pas concluants. Le diagramme d'attention ne fait pas une ligne diagonale et ça ne ressemble même pas à une ligne.

Sauf pour le dernier test, en utilisant la base de donnée finale et entraînant le modèle à partir du synthesizer entraîné en anglais, le diagramme fait bien une ligne diagonale même si sur certains diagrammes cette diagonale s'arrête au milieu.

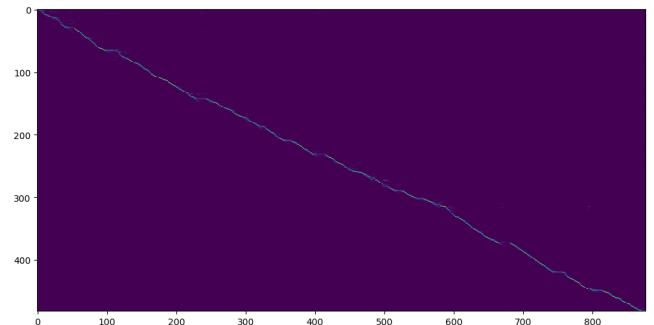
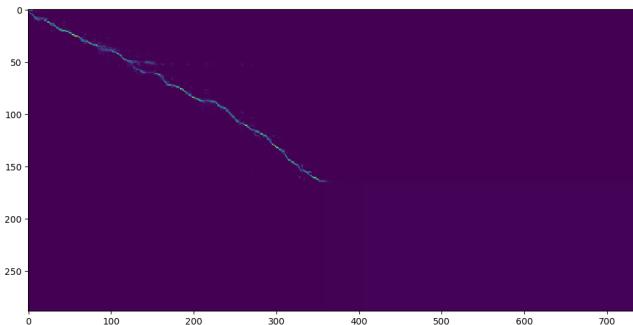


Figure 24 : image du diagramme d'attention à différentes étapes de l'entraînement à partir du synthesizer pré-entraîné (à gauche étape 316500 et à droite étape 326000)

3 - Résultats finaux

Pour les tests que ce soit avec la petite base de données ou avec la base de données finales, lors de l'entraînement à partir de rien, tous les résultats ont été un audio généré avec aucun mot audibles et la plupart du temps, un son désagréable qui devient de plus en plus fort au fil de l'audio. Les résultats n'ont pas du tout été concluants pour ces différents tests.

Les meilleurs résultats ont été obtenus avec l'entraînement à partir du synthesizer déjà entraîné en anglais. Le fait d'être parti de poids déjà déterminés et n'avoir qu'à l'entraîner pour la sonorité de la langue française a permis d'obtenir des résultats plus facilement et rapidement qu'auparavant. Sur certaines voix en entrée, l'audio généré à partir du projet est compréhensible. Et même si la voix clonée ne correspond pas grandement à la voix en entrée, elle est distinguable d'une voix masculine ou d'une voix féminine.

Les résultats sont mitigés, car le projet ne fonctionne qu'avec certains audios, des tests ont été réalisés avec des voix de personnalité, mais en fonction de la qualité de l'audio, soit la phrase dite est compréhensible soit c'est totalement incompréhensible. Et le deuxième point concerne la voix clonée, elle ne correspond pas vraiment à la voix en entrée, mais certaines caractéristiques de la voix peuvent être retrouvées si c'est une voix masculine ou féminine comme dit précédemment, mais aussi le timbre de voix si la voix est grave ou aigüe.

AXE D'AMELIORATION

Suite aux résultats mitigés obtenus lors de ce projet, certains axes d'améliorations ont été soulevés pour rendre le projet meilleur et s'approcher d'un clonage de voix presque parfait. Tout d'abord, il a manqué du temps pour réaliser un entraînement plus long sur les différents modules et également du matériel plus performant pour gagner en rapidité sur les différentes phases de l'entraînement et du prétraitement des données.

Mais l'axe d'amélioration le plus important est la base de données, car en faisant une courte comparaison entre les bases de données utilisées par CorentinJ, créateur du Github, et Google, et celle du projet, on observe que sur la partie encoder la base de données est assez complète, mais quand on s'intéresse à la partie synthesizer, la base de données utilisée dans le projet est bien insuffisante.

Base de données encoder :

Projet : Base de données Mozilla Common Voice : 16082 voix pour une durée totale de 826h

CorentinJ : Base de données train-other-500 : 1200 voix pour une durée de 500h

Google : Base de données interne avec environ 18 000 voix

Base de données synthesizer :

Projet : Base de données créée : 407 voix pour 210h d'audio

CorentinJ : Base de données train-clean-100 et train-clean-360 : 1172 voix pour 460h d'audio

Google : Utilisation de les mêmes bases de données que CorentinJ train-clean-100 et train-clean-360

Sur la partie synthesizer, il y a un écart de 800 voix entre la base de données du projet et celle utilisée par CorentinJ et Google et la base de données de CorentinJ et de Google a une durée d'audio deux fois plus longue que celle du projet. Pour obtenir de meilleurs résultats, il faudrait étoffer la base de données du synthesizer avec plus d'heures d'audio et surtout avec plus de voix différentes.

CONCLUSION

1 - Apports individuels et collectifs

Théo LOPEZ : Lors de ce projet, j'ai eu l'occasion de travailler en groupe et de manipuler différents langages de programmation. J'ai pu apprendre à gérer un projet technique sur une période de temps assez vaste. Le sujet m'a vraiment intéressé pour sa proximité avec la cybersécurité. Le fait d'être amené à développer un tel projet de A à Z est représentatif du travail que l'on pourrait réaliser en entreprise. C'est donc un exercice très enrichissant.

Yohann CHARRIER : Ce projet a été très intéressant d'un point de vue technique car on a dû s'approprier le projet de quelqu'un d'autre pour pouvoir l'utiliser pour le projet et aussi on dû utiliser des technologies qu'on n'a pas l'habitude d'utiliser. Il nous a permis de découvrir le déroulement d'un projet sur plusieurs mois avec les différents imprévus qu'il peut y avoir et devoir s'y adapter pour essayer d'amener le projet à son terme. Cela reste frustrant de ne pas avoir eu plus de temps pour essayer d'obtenir de meilleurs résultats.

2 - Conclusion générale

Après ces 10 semaines de projet. Le projet Voice cloning est donc capable de produire des voix clonés pour des discours en français. Pour perfectionner les résultats des sorties audios, il faudrait une base de données plus conséquente et plus de temps d'entraînement. En effet, vers la fin du projet les résultats étaient de plus en plus similaires à la voix originale. Il pourrait être intéressant de continuer ces travaux pour mettre en place un système comme celui-ci dans des situations réelles quand il sera opérationnel.

Aussi cliché que cela puisse paraître, nous comprenons qu'avec de grands pouvoirs vient une grande responsabilité. L'éthique et la morale sont au cœur de ce projet. Nous savions que nous devions répondre à la demande de l'enseignant encadrant du projet concernant les deep fake, c'est-à-dire les personnes se faisant passer pour quelqu'un d'autre. Avec l'augmentation des voix synthétiques, l'éthique est un problème sérieux et nous voulions partager la façon dont nous abordons ce problème à mesure que cette technologie devient courante.

BIBLIOGRAPHIE / WEBOGRAPHIE

- [1] - Site web : Le deepfake audio, la nouvelle arnaque tendance développée par les hackers, 20 Minutes, Publié le 29/07/20,
<https://www.20minutes.fr/high-tech/2831107-20200729-le-deepfake-audio-la-nouvelle-arnaque-tendance-des-hackers>
- [2] - Site web : Obama insulte Trump, le pape fait de la magie... "Complément d'enquête" explique les "deep fakes", FranceInfo, Publié le 14/02/2019,
https://www.francetvinfo.fr/economie/medias/video-obama-insulte-trump-le-pape-fait-de-la-magie-complement-d-enquete-explique-les-deep-fakes_3190141.html
- [3] Site web : « Article 226-4-1 - Code pénal », Légifrance, consulté le 01/04/2022,
https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000042193593#:~:text=Le%20fait%20d'usurper%20l,15%20000%20E2%82%AC%20d'amende
- [4] - Publication scientifique: Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis, Google Inc, Publié le 2/01/2019,
<https://arxiv.org/pdf/1806.04558.pdf>
- [5] Site web : Mel scale, Wikipédia, consulté le 01/04/2022
https://en.wikipedia.org/wiki/Mel_scale#cite_note-2
- [6] Site web : Phonème, Wikipédia, consulté le 01/04/2022
<https://fr.wikipedia.org/wiki/Phon%C3%A8me>
- [7] Blog : Evaluation of Speech for the Google Assistant, Google AI Blog, 21/12/2017,
<https://ai.googleblog.com/2017/12/>
- [8] - Site web : WaveNet: A generative model for raw audio, DeepMind, Publié le 08/10/2016
<https://www.deepmind.com/blog/wavenet-a-generative-model-for-raw-audio>

[9] - Site web : Flask (framework), Wikipédia, Consulté le 01/04/2022,

[https://fr.wikipedia.org/wiki/Flask_\(framework\)](https://fr.wikipedia.org/wiki/Flask_(framework))

[10] - Site web : Pratiques Agiles ou Méthode Cycle En V : Que Choisir ?, Groupe BPCE, Consulté le 01/04/2022,

<https://blogrecrutement.bpce.fr/pratiques-agiles-methode-cycle-v-que-choisir#:~:text=En%20cycle%20en%20V%2C%20la,d%C3%A9veloppement%20complet%20de%20la%20fonctionnalit%C3%A9>

[11] - Site web : Build fast, responsive sites with Bootstrap, Bootstrap, consulté le 01/04/2022,

<https://getbootstrap.com/>

[12] - Site web : Common Voice, Consulté 01/04/2022,

<https://commonvoice.mozilla.org/fr>

[13] - Site web :The Intuition Behind Voice Cloning with 5 Seconds of Audio, Medium, Publié le 25/12/2019,

<https://medium.com/analytics-vidhya/the-intuition-behind-voice-cloning-with-5-seconds-of-audio-5989e9b2e042>

[14] - Github : Wiki Training du projet Real-Time-Voice-Cloning, CorentinJ, pUBLIÉ le 28/12/2021

<https://github.com/CorentinJ/Real-Time-Voice-Cloning/wiki/Training>

[15] - Forum Github : Réponse de l'utilisateur NTT123, Publiée le 12 Avril 2018,

<https://github.com/keithito/tacotron/issues/144>

[16] - Publication scientifique : Tacotron, Introduced by Wang et al. in Tacotron: Towards End-to-End Speech Synthesis, Consulté le 01/04/2022

<https://paperswithcode.com/method/tacotron>

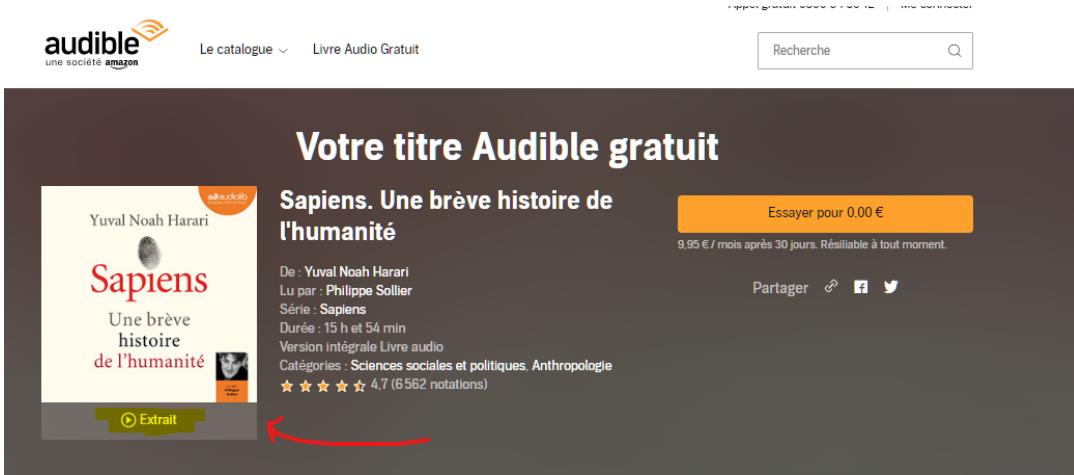
[

ANNEXES

ANNEXE 1 : Méthode de construction de la base de donnée

Etape 1 :

Trouver tous les livres audios, en français, sans musique durant l'extrait d'un lecteur en particulier (ici Philippe Sollier), Télécharger tous les extraits de ce lecteur pour faire une quarantaine de minutes en totale.



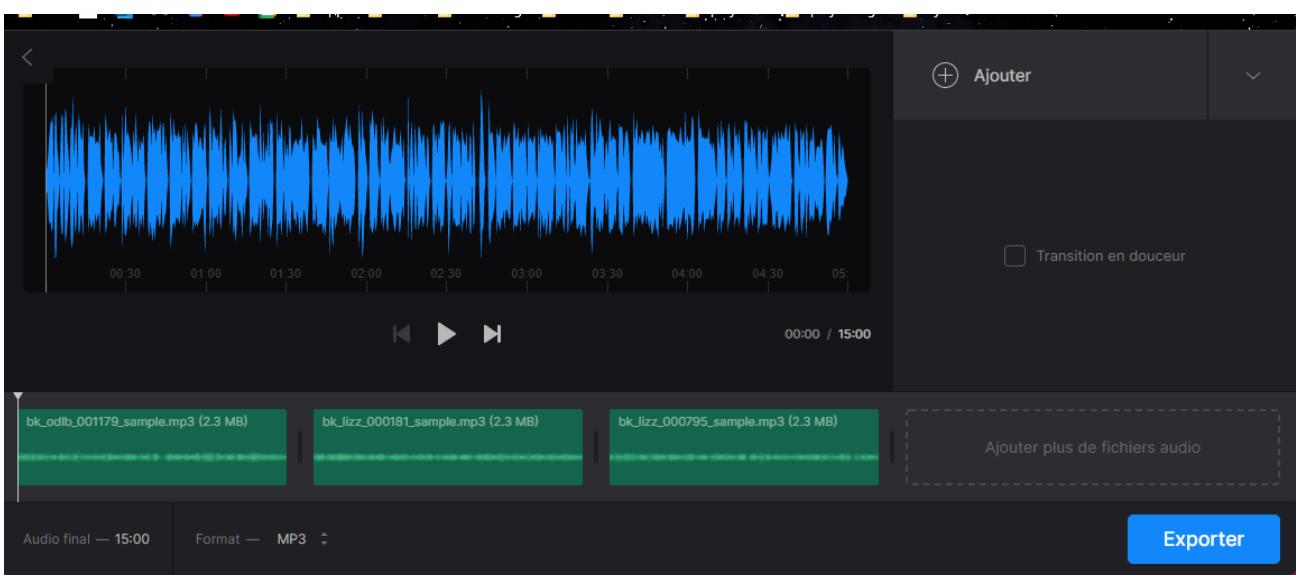
The screenshot shows the Audible website interface. At the top, there's a navigation bar with the Audible logo, a search bar, and a 'Recherche' button. Below the header, a large banner reads 'Votre titre Audible gratuit'. It features a thumbnail of the audiobook 'Sapiens. Une brève histoire de l'humanité' by Yuval Noah Harari. To the right of the thumbnail, there's a yellow button labeled 'Essayer pour 0,00 €' and a note '9,95 € / mois après 30 jours. Rémissible à tout moment.' Below the banner, product details are listed: Author: Yuval Noah Harari, Narrator: Philippe Sollier, Series: Sapiens, Duration: 15 h et 54 min, Type: Version intégrale Livre audio, Categories: Sciences sociales et politiques, Anthropologie. The rating is 4.7 (6562 notations). A red arrow points from the bottom left towards the 'Extrait' button on the book cover thumbnail.

Vous êtes membre Amazon Prime ?
Bénéficiez automatiquement de 2 livres audio offerts.
Bonne écoute !

Les auditeurs ayant acheté ce titre ont aussi aimé

Etape 2 :

Fusionner tous ces audio à l'aide d'outils sur internet ou avec ffmpeg pour fait un seul audio



The screenshot shows a digital audio workstation (DAW) interface. At the top, there's a waveform preview for a 15-minute audio clip. Below the waveform, there are three individual audio tracks listed: 'bk_0dlb_001179_sample.mp3 (2.3 MB)', 'bk_lizz_001081_sample.mp3 (2.3 MB)', and 'bk_lizz_000795_sample.mp3 (2.3 MB)'. Each track has a green waveform preview. At the bottom of the screen, there are controls for 'Format' set to 'MP3' and a large blue 'Exporter' button. The total duration of the combined audio is indicated as '00:00 / 15:00'.

Etape 3 :

Ouvrir le fichier audio sur le logiciel 360converter pour qu'il génère la transcription de l'audio, il y a les timecodes avec le texte correspondant en dessous. Quand la transcription est finie, les timecodes et le texte sont copiés dans un fichier texte.

The screenshot shows the 360converter software interface. On the left, there's a sidebar with various options: Open Local, Open YouTube, Export, Proofread, Language (French), Timestamp, Settings, How to Use, Check Update, Feedback, and About. The main window displays a transcript of a speech. Each line of text is preceded by a timestamp in the format HH:MM:SS - HH:MM:SS. The text is in French and describes a narrative about a journey and a meeting with a Mr. Monsieur. The software has a dark theme with light-colored text boxes.

Etape 4 :

Utilisation d'un script python pour générer les marqueurs et les fichiers textes de chaque timecodes. Ensuite sur Audacity on importe les marqueurs pour découper l'audio.

```
file = open('transcript.txt','r')
file2 = open('marqueur.txt','w')
line = file.readlines()

for i in range(0,len(line),2):      #lit les timecodes et les convertit en seconde
    tmp = line[i][:len(line[i])-1]  #pour réaliser des marqueurs compatible avec Audacity
    time = tmp.split(" ")[0].split(":")
    minute = time[1]
    seconde = int(minute)*60 + int(time[2])
    file2.write(str(seconde)+"\t"+str(seconde)+"\n")

for i in range(1,len(line),2):      #crée un fichier texte pour chaque timecode
    tmp = line[i][:len(line[i])-1]
    nb = int((i-1)/2)+1
    if nb<10:
        filetmp = open('out-0'+str(nb)+'.txt','w')
    else:
        filetmp = open('out- '+str(nb)+'.txt','w')
    filetmp.write(tmp)
    filetmp.close()

file.close()
file2.close()
```

ANNEXE 2 : Contrat d'accord d'utilisation de la voix dans le cadre des tests du projet M1



ALL IS DIGITAL!

OUEST



Institut Supérieur de l'Électronique
et du Numérique

Tél. : +33 (0)2.98.03.84.00

Fax : +33 (0)2.98.03.84.10

20, rue Cuirassé Bretagne

CS 42807 - 29228 BREST Cedex 2 - FRANCE

Projet M1

Année Scolaire 2021 / 2022

ACCORD D'UTILISATION

Je soussigné Monsieur ou Madame : _____, Accepte que ma voix soit utilisée dans le cadre du projet M1 2022 dirigé par Matthieu Saumard. A condition que cela reste dans le cadre scolaire. Et que ça ne soit pas utilisé à des fins malveillantes.

Date :

Signature :

ANNEXE 3 : Les exigences des versions des logiciels

Pour l'interface :

```
Flask==2.0.3  
Bootstrap==5.1.3
```

Pour le systèmes voice cloning

```
inflect==5.3.0  
librosa==0.8.1  
matplotlib==3.5.1  
numpy==1.20.3  
Pillow==8.4.0  
PyQt5==5.15.6  
scikit-learn==1.0.2  
scipy==1.7.3  
sounddevice==0.4.3  
SoundFile==0.10.3.post1  
tqdm==4.62.3  
umap-learn==0.5.2  
Unidecode==1.3.2  
urllib3==1.26.7  
visdom==0.1.8.9  
webrtcvad==2.0.10  
librosa==0.8.1  
matplotlib==3.5.1  
numpy==1.20.3  
Pillow==8.4.0  
PyQt5==5.15.6  
scikit-learn==1.0.2  
scipy==1.7.3  
sounddevice==0.4.3  
SoundFile==0.10.3.post1  
tqdm==4.62.3  
umap-learn==0.5.2  
Unidecode==1.3.2  
urllib3==1.26.7  
visdom==0.1.8.9  
webrtcvad==2.0.10  
torch  
librosa
```