AVRIL Léo, DUHAMEL Shawn, GEORGE Gabriel, GOULIN Noah et VIEILLEVIGNE Julien

Rapport final

Méd-IA

I4-PAP-2358-20240209

Publication: 09/02/2024

Contexte/ Enjeux

Les élections présidentielles génèrent de nombreuses analyses basées principalement sur les sondages et les résultats électoraux. Cependant, les analyses automatiques des débats et interviews télévisés sont moins courantes, bien qu'elles soient cruciales dans la campagne. Ce projet de R&D, en partenariat avec Alten, vise à manipuler des fichiers audios pour offrir un outil objectif et accessible aux politiques, professionnels et particuliers, enrichissant ainsi la démocratie.

Résultats

État de l'art sur les technologies avancées pour créer un pipeline qui effectue la diarisation d'un fichier audio, sépare les voix superposées, identifie les locuteurs et convertit la parole en texte.

Conclusions

Ce projet a été une réussite avec la mise en place d'un pipeline regroupant nos différents modèles d'IA. Ce projet mélangeant R&D et ingénierie nous a beaucoup appris sur les domaines de l'audio et de la recherche, thématiques trop souvent peu explorés.

Table of Contents

Abstract – Fr

Le projet Méd-IA, mené par une équipe de cinq étudiants-ingénieurs en Master 2 spécialisés en Data Science à l'EFREI, vise à révolutionner l'analyse des vidéos de débats politiques en France à travers l'usage des technologies d'intelligence artificielle. L'objectif principal est d'approfondir la compréhension de ces débats en identifiant les thèmes abordés, les émotions exprimées, et le temps de parole des participants, afin de contribuer à une démocratie plus participative.

Le projet s'appuie sur un soutien technique et matériel fourni par l'entreprise Alten, incluant un dataset de bandes audios et des connaissances approfondies en matière d'état de l'art sur des algorithmes avancés. L'équipe a utilisé Python et des bibliothèques de pointe telles que TensorFlow et PyTorch pour le développement de la solution technique, préférant cette dernière pour sa pertinence dans le domaine de la recherche.

La solution technique développée comprend plusieurs composants clés, notamment la diarisation de l'audio, la séparation des voix superposées, l'identification des locuteurs, et la transcription en texte, formant ensemble un pipeline complexe mais efficace pour le traitement et l'analyse des données audio.

L'équipe a rencontré divers défis liés à la complexité du traitement audio, mais a réussi à surmonter ces obstacles grâce à une collaboration efficace et une méthodologie agile. Le projet a également bénéficié de la mise en œuvre de modèles d'IA à la pointe de la technologie et d'une infrastructure technique solide.

En conclusion, le projet Méd-IA marque une avancée significative dans l'analyse automatisée des discours politiques, offrant des perspectives prometteuses pour l'amélioration de la participation démocratique et la compréhension des dynamiques politiques en France. Les résultats obtenus et les expériences acquises par l'équipe constituent une base solide pour de futures recherches et développements dans ce domaine.

Abstract – Eng

The Méd-IA project, led by a team of five Master's degree students specialized in Data Science at EFREI, aims to revolutionize the analysis of political debate videos in France through the use of artificial intelligence technologies. The main objective is to deepen the understanding of these debates by identifying the topics discussed, the emotions expressed, and the speaking time of the participants, to contribute to a more participatory democracy.

The project relies on technical and material support provided by the company Alten, including a dataset of audio recordings and in-depth knowledge of state-of-the-art algorithms. The team used Python and cutting-edge libraries such as TensorFlow and PyTorch for the development of the technical solution, preferring the latter for its relevance in the research domain.

The developed technical solution comprises several key components, including audio diarization, separation of overlapping voices, speaker identification, and transcription into text, together forming a complex yet effective pipeline for the processing and analysis of audio data.

The team encountered various challenges related to the complexity of audio processing but managed to overcome these obstacles through effective collaboration and agile methodology. The project also benefited from the implementation of state-of-the-art AI models and a solid technical infrastructure.

In conclusion, the Méd-IA project represents a significant advancement in automated analysis of political discourse, offering promising prospects for improving democratic participation and understanding political dynamics in France. The results obtained and the experiences gained by the team provide a solid foundation for future research and development in this field.

Introduction

Notre projet vise à analyser les vidéos de campagne présidentielle en France en utilisant des technologies d'intelligence artificielle. Dans un contexte où les analyses politiques traditionnelles reposent largement sur les sondages et les résultats électoraux, notre approche se veut novatrice en permettant une meilleure compréhension et analyse des débats politiques, notamment en identifiant les thèmes abordés, les émotions exprimées et le temps de parole de chaque participant.

Le système que nous développons aura pour objectif d'offrir une vision plus approfondie des discours politiques, ouvrant ainsi la voie à une démocratie plus participative et permettant un décodage plus précis du langage du pouvoir politique. Cette initiative répond à un besoin crucial dans le paysage politique français, où les analyses automatiques des débats politiques et des interviews sont encore peu développées malgré leur importance indéniable en période de campagne électorale.

L'objectif principal de ce document est de présenter de manière claire et concise notre projet, ses enjeux et ses objectifs, tout en informant le lecteur sur les ressources disponibles telles que le glossaire et les références, facilitant ainsi sa compréhension du document.

Le lecteur est encouragé à se référer aux sections suivantes pour une analyse détaillée du contexte du projet, de sa description, des contributions des acteurs impliqués, ainsi que des conclusions et perspectives pour l'avenir du projet.

Nous vous conseillons d’aller voir les définitions importantes dans l’annexe « Mots clés » pour mieux comprendre notre sujet.

Remerciements

Nous tenons à remercier tout particulièrement Yves Soronellas et Zineb Noumir, notre mentor et déposante du sujet pour leurs précieuses aides tout au long du projet. Sans eux nous n’aurions pas pu aller aussi loin dans Méd-IA.

I) Contexte

Dans cette section, nous plongeons dans le contexte complexe entourant les élections présidentielles en France, mettant en lumière les défis et les opportunités dans le domaine de l'analyse politique.

Étude du Type d'Utilisateurs et de Leurs Besoins

Les utilisateurs potentiels de notre système varient, des chercheurs et analystes politiques aux citoyens engagés. Leur besoin commun est une analyse approfondie des discours politiques pour une compréhension accrue des enjeux politiques. Ces utilisateurs recherchent des outils qui leur permettent d'aller au-delà des résultats électoraux pour saisir les nuances des discours et des débats politiques.

Dans ce contexte, les attentes des utilisateurs s'articulent autour de plusieurs axes. Ils cherchent des outils qui leur permettent de décoder le langage politique complexe, d'identifier les thèmes récurrents dans les discours des candidats, d'évaluer l'impact émotionnel des interventions politiques, et d'obtenir une répartition équitable du temps de parole entre les différents acteurs politiques.

Étude du Contexte Technico-fonctionnel Existants

Traditionnellement, les analyses politiques se concentrent sur les sondages et les résultats électoraux, reléguant souvent les discours politiques et les interviews à l'arrière-plan. Cette lacune souligne le potentiel d'une approche innovante utilisant des technologies d'intelligence artificielle pour analyser systématiquement et en profondeur ces discours.

Dans le contexte technico-fonctionnel actuel, l'automatisation de l'analyse politique est encore à ses balbutiements. Des initiatives prometteuses émergent, mais la plupart des outils existants se concentrent sur des aspects limités de l'analyse, laissant de nombreux besoins non satisfaits. Il est crucial de reconnaître que notre projet s'inscrit dans un écosystème où l'automatisation de l'analyse politique est encore en phase de développement, offrant ainsi une opportunité unique de contribuer à son avancement et à son enrichissement.

Positionnement dans le Marché et État de l'Art

L'analyse automatique des débats politiques en France est un domaine en évolution. Bien que des outils similaires existent, tels que DISPUTool aux États-Unis, leur adaptation au contexte politique français demeure un défi. Notre projet se distingue par sa focalisation sur le paysage politique français, offrant une approche spécifique et adaptée aux besoins du public français.

Dans un marché où la demande pour des analyses politiques automatisées est croissante, notre projet s'inscrit comme une réponse pertinente aux attentes des acteurs politiques et de la société civile. Il offre une perspective unique et une valeur ajoutée grâce à notre engagement spécifique dans le contexte politique français.

1. Enjeux

Notre projet vise à ouvrir la voie à une démocratie plus participative en permettant une analyse approfondie des discours politiques. Il s'agit de décoder le langage du pouvoir politique pour une compréhension plus éclairée des enjeux politiques. En fournissant des outils et des analyses accessibles, notre projet cherche à responsabiliser les citoyens et à encourager un engagement politique plus informé.

Concurrence et Opportunités

DISPUTool, notre principal concurrent, s'intéresse également aux débats politiques français. Cependant, notre projet offre une perspective unique grâce à notre engagement spécifique dans le contexte politique français. Cette concurrence souligne l'importance et la pertinence de notre initiative dans un paysage politique en évolution constante.

En capitalisant sur nos forces et en tirant parti des opportunités émergentes, nous sommes bien positionnés pour apporter une contribution significative à l'analyse politique automatisée en France. Notre projet aspire à être un catalyseur de changement dans la manière dont nous comprenons et engageons avec la politique, ouvrant ainsi la voie à une démocratie plus participative et inclusive.

En résumé, notre projet représente une avancée significative dans le domaine de l'analyse politique en France. En répondant aux besoins d'une démocratie plus participative et en relevant le défi de la concurrence avec des outils déjà établis, nous aspirons à être des pionniers dans le domaine de l'analyse politique automatisée en France.

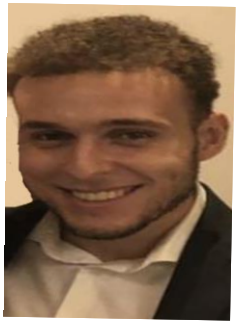
Fonctions et Contributions des Acteurs

Dans cette section, nous mettons en lumière notre rôle essentiel dans le développement du projet ainsi que nos contributions spécifiques à sa réalisation.

Notre Équipe

Nous formons une équipe dynamique de cinq étudiants-ingénieurs en Master 2 spécialisés en Data Science. Chacun d'entre nous apporte une expertise distincte et complémentaire, ce qui enrichit notre collaboration et renforce notre capacité à relever les défis du projet.

Nos Contributions Individuelles

				
Léo	Julien	Gabriel	Noah	Shawn
Chargé de la diarisation de locuteur, de la séparation de voix, et de la mise en place des pipelines de traitement des données. Léo veille à la structuration des enregistrements audio des débats politiques pour garantir une analyse précise.	Spécialisé dans la diarisation de locuteur et la transcription, Julien identifie les intervenants et transcrit leurs discours avec exactitude pour fournir des données fiables pour l'analyse politique.	Responsable de l'identification de locuteur et de la gestion des pipelines de traitement des données. Gabriel développe des algorithmes pour organiser le flux de données et assurer leur traitement efficace.	En charge de l'identification de locuteur, de la transcription des enregistrements audio, et de la gestion des pipelines. Noah garantit l'exactitude et l'intégrité des données utilisées dans l'analyse politique.	Responsable de la diarisation de locuteur et de l'identification des intervenants. Shawn assure une analyse précise du temps de parole et des interactions lors des débats politiques.

Notre Motivation et Nos Engagements

Notre équipe est unie par une passion commune pour l'exploration des données et l'application de techniques d'intelligence artificielle à des problèmes complexes. Nous sommes engagés à relever les défis du projet avec détermination et créativité, inspirés par l'opportunité de contribuer à l'avancement de l'analyse politique automatisée.

Notre Gestion de Projet et Notre Méthodologie

Pour assurer le succès du projet, nous avons adopté une approche agile, basée sur des sprints de deux semaines. Nous organisons régulièrement des points d'avancement avec le Product Owner (PO), favorisant ainsi une communication transparente et une adaptation rapide aux exigences du projet.

Outils Utilisés

Nous utilisons des outils technologiques avancés pour faciliter notre collaboration et notre gestion de projet. Des plateformes telles que Discord pour la communication en équipe et Teams pour les réunions et le partage de documents garantissent une coordination efficace et une organisation optimale des tâches.

Cette combinaison d'expertise, de méthodologie agile et d'outils technologiques nous positionne favorablement pour atteindre nos objectifs et produire des résultats de haute qualité dans le cadre de notre projet d'analyse politique automatisée.

II) Solution technique

1. Soutien d'Alten

L'entreprise déposante du sujet Alten nous a beaucoup conseillé tout au long du projet et nous a fourni dataset et code.

Concernant le dataset, nous avons eu des bandes audios de 10 secondes de type WAV pour 18 hommes politiques. C'est ce dataset que nous allons utiliser plus tard pour l'identification des locuteurs.

Alten nous a aussi fourni du code à travers des algorithmes partagés via Teams. Voici la synthèse de leurs travaux :

Pour l'embedding (afin de passer de données audios à des vecteurs numériques), Alten a utilisé xVector avec PyTorch puis UBM (Universal Background Model) et iVector.

Ensuite vient le processus de clustering pour identifier et séparer différentes voix présentes. En règle générale, l'UBM est un modèle qui est utilisé pour modéliser les caractéristiques de la voix qui sont communes à tous les orateurs.

L'iVector a pour objectif de réduire les caractéristiques de haute dimension pour avoir un vecteur plus faible qui garde l'information essentielle pour distinguer les locuteurs. iVector utilise le modèle UBM comme référence. Les deux modèles sont liés.

Le xVector est une approche plus récente et complexe qui utilise les réseaux de neurones qui sont entraînés pour apprendre à distinguer les orateurs sans utiliser UBM (séparation de voix).

Ces modèles sont utilisés pour la séparation des voix superposées.

Ces travaux décris ci-dessus nous ont donné un aperçu du travail que nous devrions faire et des améliorations à effectuer.

Alten nous a aussi fourni une machine virtuelle nous permettant d'entraîner nos modèles d'IA plus rapidement grâce au GPU mais nous ne l'avons pas utilisé car nous avions eu des problèmes pour s'y connecter et parce que nos modèles n'en avaient pas spécialement besoin.

2. Outils pour faire de l'IA

Pour réaliser notre projet de R&D Méd-IA, nous avons décidé de nous concentrer sur ce que nous maîtrisons le mieux, soit le langage de programmation Python. Ce langage est très populaire dans le domaine de la data science et de l'intelligence artificielle. Nous pouvons retrouver sur Python les deux bibliothèques les plus populaires pour faire de l'IA Deep Learning : TensorFlow de Google et PyTorch de Meta.

TensorFlow est une bibliothèque de Python que l'on a l'habitude d'utiliser pour presque tous nos projets et nos cours de Deep Learning, tandis que PyTorch est une bibliothèque qui n'a jamais ou peu été utilisée par notre équipe.

L'avantage d'utiliser TensorFlow pour ce projet est que nous connaissons déjà le cadre de travail, ce qui nous facilitera la mise en place des modèles.

TensorFlow est connu pour être axé sur l'industrie tandis que PyTorch est axé sur la recherche, ce qui est plus ce que l'on recherche à ce stade du projet.

Pour faire un choix sur la bibliothèque à choisir, nous devons prendre en compte la disponibilité des modèles, le déploiement de l'infrastructure et l'écosystème.

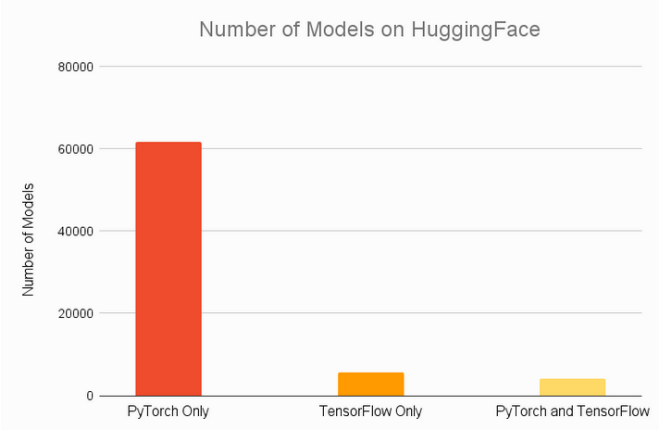
Le déploiement de l'infrastructure nous importe peu, donc nous allons regarder de plus près la disponibilité des modèles et l'écosystème.

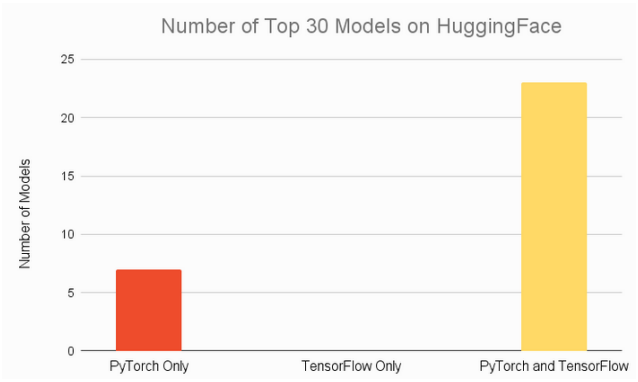
PyTorch est clairement à privilégier pour la recherche en raison du nombre de modèles disponibles, de la facilité à ajuster les modèles et du nombre de publications de recherche utilisant PyTorch. Cependant, TensorFlow reste un très bon candidat pour faire de la recherche en raison de la disponibilité des modèles les plus populaires sur TensorFlow.

Durant ce projet, nous allons privilégier PyTorch mais nous pourrions aussi utiliser TensorFlow.

De plus, presque tous les modèles pré-entraînés par les grandes entreprises tech que nous allons utiliser ont été développés en utilisant PyTorch. Il est ainsi indispensable de connaître cette bibliothèque et de savoir l'utiliser pour ce type de projet.

Vous retrouverez ci-dessous les deux graphgiques représentant le nombre de modèle selon TensorFlow et PyTorch.





III) Conception du projet

Diarisation

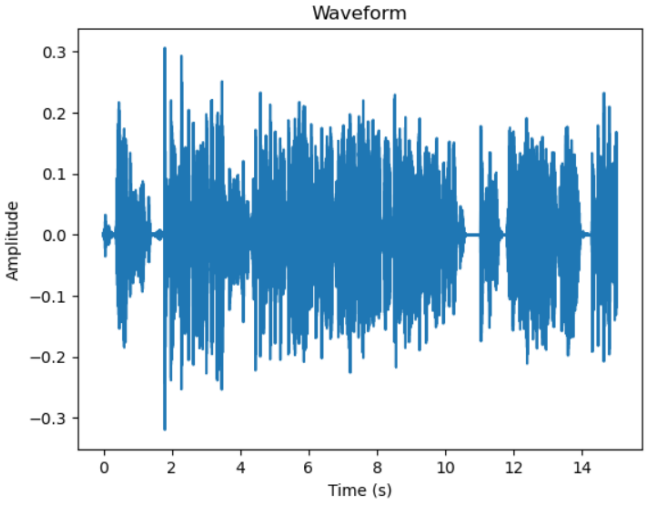
La diarisation du locuteur est la tâche de déterminer automatiquement qui parle et quand, à travers un fichier audio ou dans une vidéo (Télécom SudParis).

Pour réaliser cette diarisation, nous avons regardé les dernières technologies et outils sortis sur ce sujet. Vous trouverez ci-dessous le tableau comparatif de Pyannote de HuggingFace et NeMo de NVIDIA. Les deux sont des outils qui font plus ou moins la même chose, sauf que Pyannote détecte mieux lorsque des personnes parlent en même temps. C'est cette spécificité qui le rend plus intéressant, mais il en a aussi d'autres, comme la détection automatique de la personne qui parle et le fait que les modèles pré-entraînés sont accessibles en open-source.

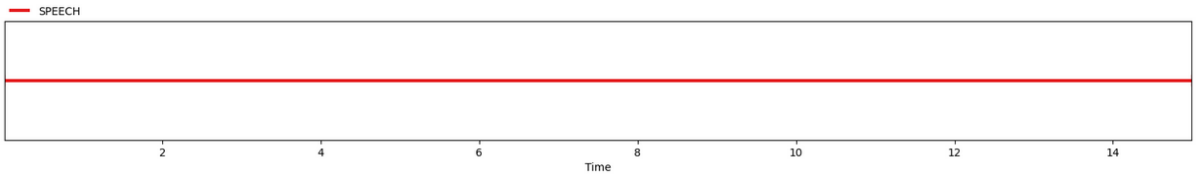
	pyannote	Nemo
Pre-trained models available	✅	✅
Good overlapping speakers detection (multilabel segmentation)	✅	➖
Easy integration with ASR task and downstream NLP tasks	➖	✅
Possibility to specify the number of speaker as a parameter for inference	✅	✅
Automatic detection of the number of speakers	✅	✅
Models available for specific use cases (phone call, outdoor conversation, high quality,...)	❌	✅
Highly customizable pipeline	➖	✅

Il faut savoir que Pyannote s'appuie sur une pipeline impliquant plusieurs modèles pour le traitement de l'audio. Il y a par exemple un modèle pour détecter les discussions, un modèle pour détecter lorsque des personnes parles en même temps, un modèle pour savoir qui parle et quand et pleins d'autres modèles.

Pour illustrer cela, nous avons pris 15 secondes d'un débat entre Emmanuel Macron et Marine Lepen. Vous retrouverez notre fichier audio à droite.



Via un modèle de Pyannote ,pour la détection de voix, nous avons détecté que l'on a 15 secondes de débat sans interruption.

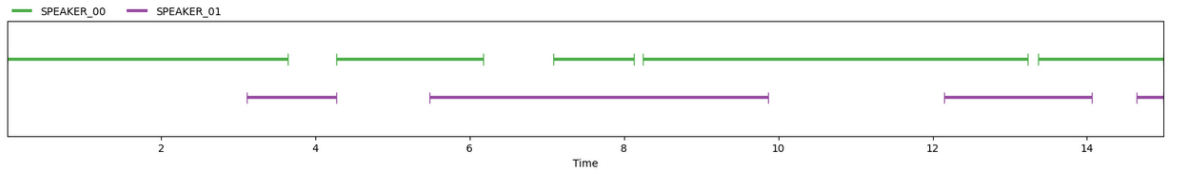


Notre diarisation fonctionne plutôt bien même si elle n'est pas parfaite. Pyannote permet de faire de la détection de la parole, de savoir quel interlocuteur parle et quand, et de détecter les superpositions de voix.

Comme vous pouvez le voir ci-dessous, nous savons qui parle et quand avec précisement quand Macron (Speaker 0) ou Lepen (Speaker 1) comencent à parler et termine de parler.

```
start=0.0s stop=3.6s speaker_SPEAKER_00
start=3.1s stop=4.3s speaker_SPEAKER_01
start=4.3s stop=6.2s speaker_SPEAKER_00
start=5.5s stop=9.9s speaker_SPEAKER_01
start=7.1s stop=8.1s speaker_SPEAKER_00
start=8.2s stop=13.2s speaker_SPEAKER_00
start=12.1s stop=14.1s speaker_SPEAKER_01
start=13.4s stop=15.0s speaker_SPEAKER_00
start=14.6s stop=15.0s speaker_SPEAKER_01
```

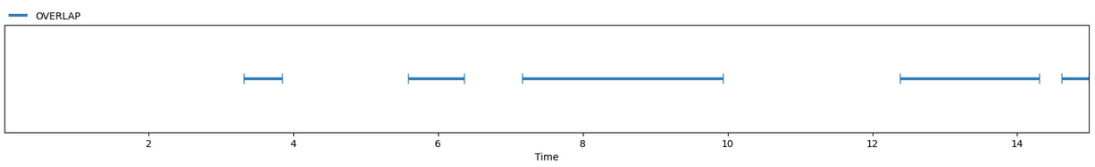
```
1 diarization
```



Nous pouvons aussi d'avoir en utilisant un autre modèle de pyannote quand est ce que les locuteurs parlent en même temps

```
[ 00:00:03.319 -->  00:00:03.848]
[ 00:00:05.588 -->  00:00:06.356]
[ 00:00:07.158 -->  00:00:09.940]
[ 00:00:12.380 -->  00:00:14.308]
[ 00:00:14.616 -->  00:00:14.991]
```

Out[3]:



Le problème qui va se poser à nous maintenant est celui de séparer les voix car Pyannote n'est pas capable de le faire. Pour cela, nous avons essayé d'utiliser recognize_google de Speech Recognition Google afin de faire du speech to text mais cela ne fonctionne pas lors de superpositions de voix malgré les informations de qui parle quand durant ces superpositions.

Nous avons aussi pensé à tester le modèle Spleeter de Deezer qui permet de séparer les voix des musiques. À voir s'il fonctionnerait lors de superpositions de voix. C'est une piste que nous n'avons pas encore essayée mais d'autres idées restent à trouver.

Séparation de voix

La séparation des voix est le deuxième point technique clé de la réussite de notre projet. C'est une discipline complexe qui consiste à estimer à partir d'un signal audio mélangeant plusieurs voix, les sources individuelles qui le compose. La complexité de la tâche réside notamment dans le fait que bien souvent dans des scénarios réels comme les débats, les voix se chevauchent ce qui compliquent la séparation. L'équipe d'ALTEN rapporte qu'un débat est composé en moyenne de 12,6 % d'overlap. La complexité réside également et principalement dans la nature même les caractéristiques des voix humaines qui peuvent avoir différents tons, différents rythmes qui dépendent du temps. Le discours, est une suite de phonèmes qui sont prononcés l'un après l'autre dont l'ordre est crucial pour en comprendre le sens. Ce type de données sont caractérisées de séquentielles et demandent des modèles spécifiques pour être traitées. Les modèles utilisés doivent être capable de créer un contexte, de créer des liens entre les éléments de la séquence.

Les Recurrent Neural Networks (RNN) :

Le RNN est un modèle de deep learning qui permet de traiter une séquence de données d'une longueur quelconque étape par étape en utilisant les résultats de l'étape *t* comme entrée pour l'étape *t+1*. Cette technique permet de prendre en compte tous les éléments historiques de la séquence afin de prédire un résultat. Néanmoins, le modèle souffre des problèmes de disparition et d'explosion du gradient, c'est-à-dire que l'influence des premiers éléments de la séquence diminue exponentiellement plus la séquence est grande ce qui rend très difficile de capturer les dépendances lointaines entre les éléments de la séquence. De plus, les calculs sont lents, le processus étant réalisé de manière séquentielle.

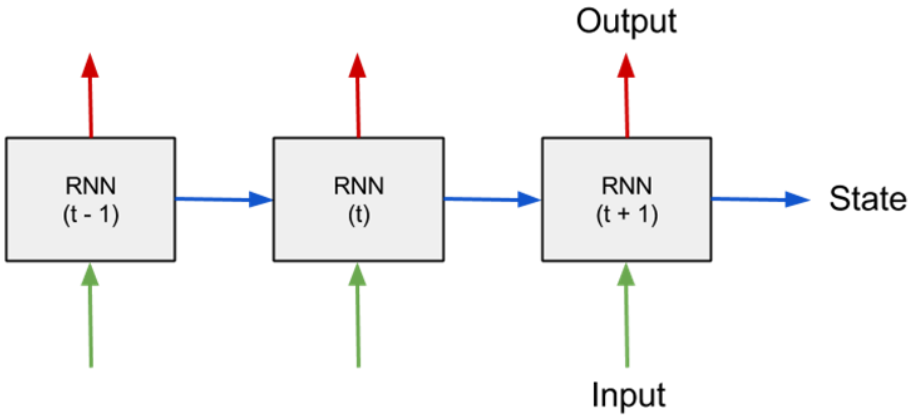


Figure 1 Architecture du modèle RNN

Les réseaux Long Short-Term Memory (LSTM) :

Le LSTM est un type spécial de RNN qui résout les problèmes de disparition et d'explosion du gradient en utilisant deux canaux distincts pour représenter les dépendances lointaines (long-term

memory) et les dépendances proches (short-term memory). En utilisant un système de porte et de cellules de mémoire, le LSTM est efficace pour capture les dépendances lointaines. Néanmoins, la lenteur des calculs persiste, le processus étant toujours séquentiel.

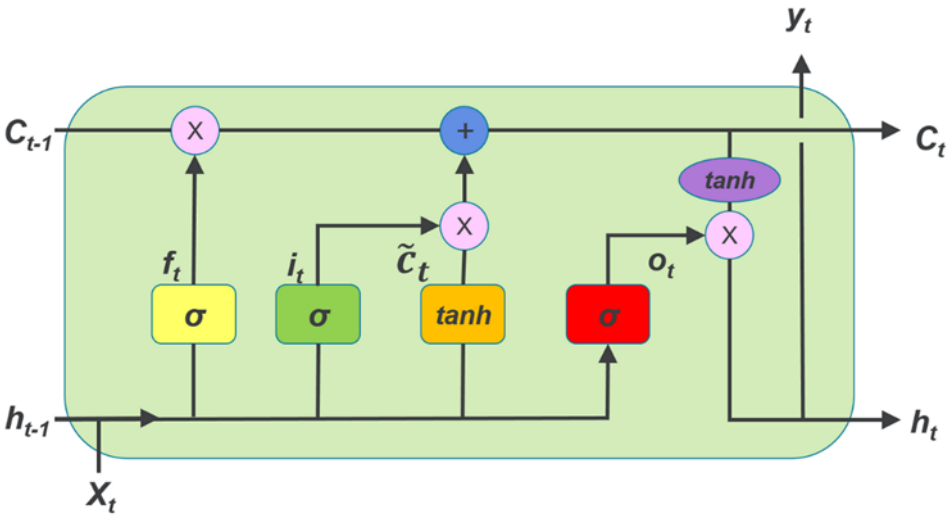


Figure 2 Architecture d'un réseau LSTM

Les transformers :

Les RNN et réseaux LSTM ont toujours été très populaire dans le domaine du traitement du langage. Cependant, aujourd'hui, les modèles qui performent le mieux sur la séparation de voix sont les transformers. Architecture introduite en 2017 par l'équipe de recherche Google Brain, elle est à l'origine d'un nouveau paradigme sur le traitement de données séquentielles. Cette architecture révolutionnaire est à l'origine des outils tels que Chat GPT. Les transformers introduisent le mécanisme de self-attention qui permet de traiter une séquence de données de longueur quelconque et de capturer les dépendances et les relations entre les éléments de la séquence d'entrée plus efficacement qu'auparavant. Les transformers n'utilisent pas de connections récurrentes comme les RNN ce qui permet de paralléliser le traitement des données ce qui augmente grandement la rapidité des calculs.

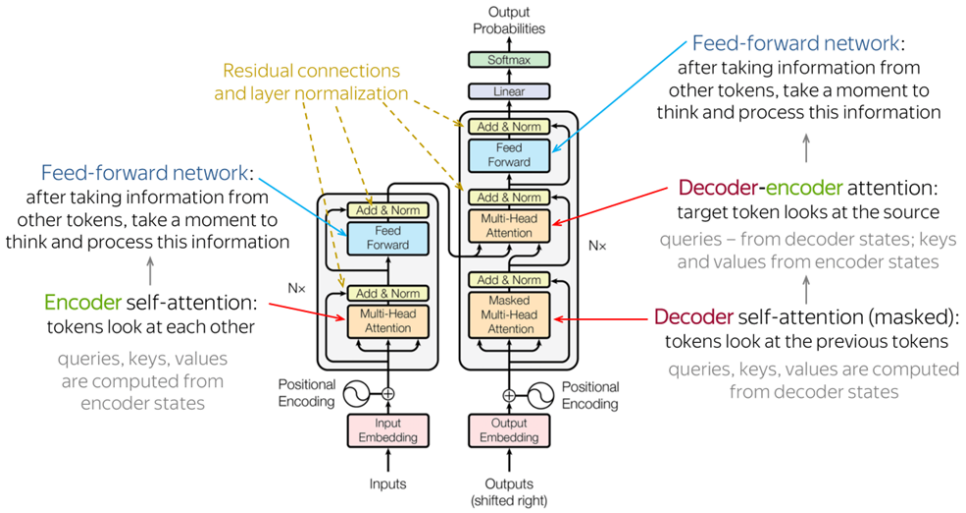


Figure 3 Architecture transformer détaillée

Depuis les travaux de Google ayant donné naissance au modèle de séparation de voix SepFormer, plusieurs autres modèles se basant sur la méthode dual-path ont vu le jour. Cette approche vise à résoudre les problèmes d'efficacité rencontrés pendant le traitement de longues séquences. Elle divise la longue séquence en petits segments et les traite par deux étapes alternées :

- Intra-processing : Concentré sur le traitement à l'intérieur de chaque segment.
- Inter-processing : Concentré sur la combinaison des informations entre les segments.

Néanmoins, cette méthode présente toujours un problème de charge de traitement due aux segments qui se chevauchent. Le modèle MossFormer 2 développé par Alibaba Group tente de résoudre le problème et présente à date les meilleures performances en matière de séparation de voix. MossFormer 2 évite les problèmes d'efficacité de la méthode dual-path en utilisant des segments non superposés et en appliquant le mécanisme de self-attention sur toute la séquence. De plus, en plus d'appliquer ce mécanisme, le modèle possède un bloc Recurrent qui lui permet de mieux capturer les dépendances proches.

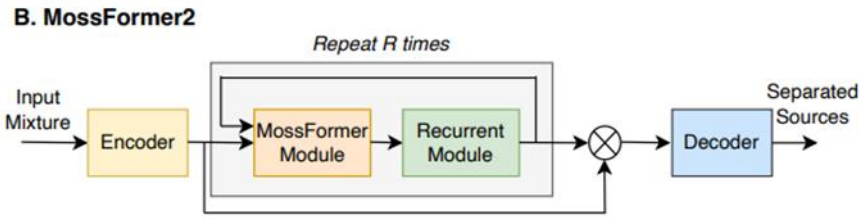


Figure 4 Architecture haut niveau de MossFormer

Identification de locuteur

Le cerveau humain est bien fait, lorsque nous entendons une voix (déjà entendu auparavant), il nous est très simple d'attribuer un visage, ou du moins une identité à cette voix. Ce réflexe automatique vient de nos nombreuses années d'expériences de vie et nous est commun à tous (ainsi qu'à de nombreux animaux). En revanche, lorsque nous décomposons ce phénomène à une échelle scientifique, cela devient tout de suite un problème très complexe qui pour vulgariser se résume en la capacité à calculer les différences entre les phonétiques incidentelles d'une vocalisation spécifique et les représentations phonologiques abstraites des mots que contient la vocalisation[1].

L'identification de locuteur est une partie importante de notre projet MED-IA. En effet, après avoir réussi à segmenter l'audio d'un débat entre les différents participants et séparer les voix lors de superpositions multiples présentes dans un débat, il faut réussir à attribuer un nom à ces orateurs. Alors bien qu'en tant que humains nous n'ayons pas besoin de cette technologie pour savoir qui parle lors d'un débat, nous avons pour but de faire des analyses personnalisées sur le débat pour chaque orateur. Dans ce cas, il nous faut être capable de savoir automatiquement qui a prononcé quelles paroles tout du long de l'audio, et pour cela il n'est pas question de le faire à la main.

Jeu de données

Afin de réaliser à bien notre modèle d'identification de politiciens, Alten nous a fournis leur jeu de données constitué de 4585 fichiers audio de type .wav. Chaque audio contient un monologue de 10 secondes d'un politicien, accompagné de son nom. Etant des audio avec des monologues sans interruption et d'une bonne clarté sonore, nous avons eu accès à des données de haute qualité. En revanche, lors d'un vrai débat, les voix ne sont pas souvent aussi claires et se chevauchent très souvent à cause d'interruptions. Il faut donc noter que même si notre modèle performe bien sur les données d'Alten, en cas réel il aura un peu plus de mal avec nos audio d'overlaps séparé.

Etat de l'art

De nos jours, pour atteindre les meilleurs résultats lors de la classification d'un fichier audio, il est crucial de respecter une certaine étape qui est l'**embedding**. En français cela se traduit littéralement par « intégration », il faut comprendre cela comme une vectorisation de l'audio. Pour aller plus loin, il est possible de représenter un audio brut directement sous forme de « vecteur de caractéristiques brutes », en effet, un fichier audio est avant tout une onde, et il nous faut être capable de la représenter numériquement pour l'exploiter.

Pour cela, il faut d'abord échantillonner notre audio qui est une onde en continu afin de pouvoir les organiser dans un vecteur. Dans notre cas, notre échantillonnage est d'une fréquence de 16 kHz ce qui veut dire que chaque seconde d'audio sera représentée par un vecteur de 16000 points.

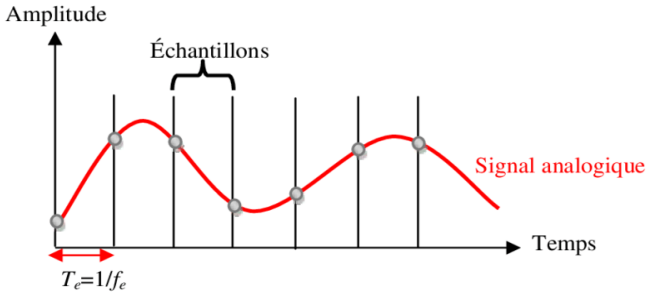


Fig1 : Echantillonnage d'une onde

Ce vecteur de caractéristiques brutes en résultant est très basique, il ne capture que l'amplitude de l'onde audio à divers points dans le temps et ne tient pas compte de caractéristiques plus complexes comme la fréquence ou le timbre. En principe, il serait suffisant pour entrainer un modèle de classification de locuteurs, néanmoins, il prendrait beaucoup de temps à entrainer en raison du nombre de valeurs et ne donnerait pas des résultats satisfaisants. Une grosse partie de la réflexion autour de l'identification de locuteurs aura été autour du choix de modèle pour générer le meilleur embedding possible, qui serait capable de capturer des relations complexes et non-linéaires de l'audio. Alten utilise deux modèles alternatifs, le I-Vector qui représente un audio en utilisant un vecteur de caractéristiques statistiques se basant sur des modèles de mélange gaussien, ainsi que le X-Vector, un modèle plus moderne basé sur les réseaux neuronaux profonds (deep neural networks), et sur lequel nous reviendrons en détail plus tard.

Nous avons passé beaucoup de temps sur le modèle wav2vec 2.0[2] développé par Facebook AI en 2020 et qui était de loin le meilleur modèle à sa sortie. Le problème de ce modèle que nous avons récupéré avec la librairie torchaudio de PyTorch est qu'il est spécialisé dans « l'Automatic Speech Recognition » (reconnaissance vocale automatique), nous induisant en erreur. En effet, nous avons cru pendant un certain temps que comme le titre peut laisser croire l'ASR était conçu pour reconnaître la parole d'un orateur. En vérité il s'agit de la transcription d'un audio en texte, expliquant pourquoi nous nous retrouvions avec des vecteurs 4D augmentés à environ 9 197 568 valeurs pour un simple audio de 10 secondes. C'est d'autant plus logique car la transcription en texte requiert beaucoup plus d'informations sur l'audio que pour effectuer une détection de locuteurs. Comme vous pouvez vous en douter, nos ordinateurs n'étaient pas assez puissants pour supporter une telle quantité de donnée et il était impossible de continuer sur cette piste (nous sommes même allés jusqu'à demander un accès à une machine virtuelle d'Alten pour faire tourner le modèle mais il y a eu des problèmes d'accès).

Dans le cadre de l'identification des locuteurs, nous avons exploré l'utilisation de NVIDIA NeMo, une plateforme open source spécialisée dans la reconnaissance vocale et le traitement du langage naturel. Cependant, nous avons rapidement rencontré des difficultés liées à l'adaptation de NeMo pour l'identification des locuteurs dans un contexte de débats politiques. En effet, Nemo est principalement utilisé pour de la vérification de locuteur, afin de s'assurer que la personne qui parle est bien celle qu'elle prétend être. Malgré nos efforts pour entraîner un modèle basé sur NeMo, nous avons constaté une complexité significative dans l'adaptation de cette plateforme à nos besoins spécifiques. Bien que nous ayons réussi à obtenir une précision de l'ordre de 65% sur notre ensemble de données, cette performance était en deçà de nos attentes, surtout compte tenu du fait que le modèle ne montrait pas une amélioration significative avec un entraînement supplémentaire. Cette limitation nous a conduit à reconsidérer notre approche technologique pour l'identification des locuteurs, en recherchant des solutions plus adaptées à notre problème spécifique.

Fort de notre succès avec la librairie Pyannote dans les domaines de la diarisation, nous avons une nouvelle fois sollicité leur expertise pour trouver un modèle d'embedding audio. Et en effet, ils en avaient bien un à proposer le modèle « embeddings ». Spécialisé dans l'embedding d'audio pour la classification (que ce soit de locuteurs, d'émotions, de langues …), ce modèle propose un embedding très léger et d'une performance inégalé grâce à la fusion de deux modèles différents, le SincNet et le X-Vector.

Pyannote embeddings

Le modèle embeddings de pyannote est un modèle d'apprentissage profond (deep learning), écrit avec la librairie PyTorch et basé sur une architecture appelée XVectorSincNet. Pour tout audio traité avec ce modèle, nous avons en sortie un vecteur 1D avec seulement 512 valeurs. Nous allons voir maintenant comment il est possible d'avoir si peu de valeurs en résultat et atteindre des performances au sommet de l'état de l'art en matière de détection de locuteurs.

SincNet

Le principal atout différenciant notre modèle de celui utilisé par Alten est l'utilisation d'un modèle SincNet[3]. Ce modèle est basé sur des fonctions Sinc, conçus spécifiquement pour s'adapter aux signaux audios, et qui implémentent des filtres passe-bande. Ce sont des dispositifs laissant seulement passer les fréquences d'un certain intervalle compris entre une fréquence de coupure basse et une fréquence de coupure haute. En physique, lorsque nous appliquons une transformée de Fourier à une onde nous la transformons d'un domaine temporel à un domaine fréquentiel. Cette opération mathématique nous permet de visualiser de manière plus intuitive ainsi que d'analyser les différentes fréquences qui composent le signal sonore.

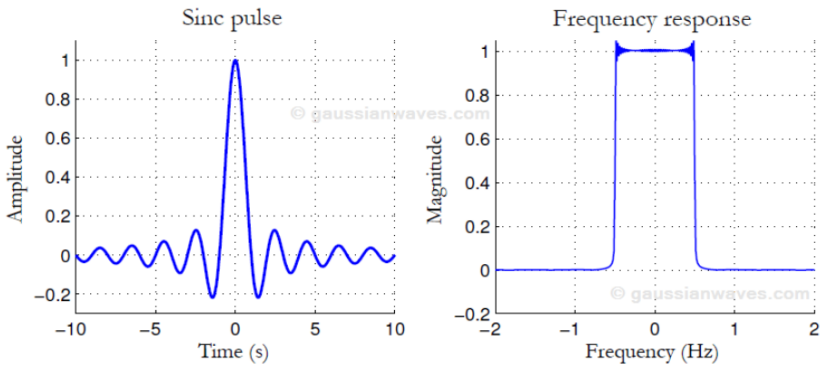


Figure 1: Sinc pulse and its manifestation in frequency domain

Fig2 : Fonction Sinc - Domaine Temporel & Fréquentiel

La première couche du modèle SincNet que nous appliquons en entrée à notre signal sonore, permet par un apprentissage automatique de trouver les meilleurs filtres passe-bande à appliquer, en ajustant les paramètres du filtre qui sont seulement les fréquences de coupure, c'est-à-dire le début et

la fin du rectangle dans le domaine fréquentiel. Evidemment le signal sonore étant continu, il y a plusieurs filtres passe-bande à appliquer du début de l'audio jusqu'à sa fin, on appelle cela un banc de filtres. Tout cela permettra d'extraire des caractéristiques plus précises en se concentrant seulement sur des bandes de fréquences spécifiques plutôt que l'audio entier. Par exemple, un filtre pourrait avoir des fréquences de coupure correspondant à la gamme de fréquences produites par les voyelles, ou des consonnes…

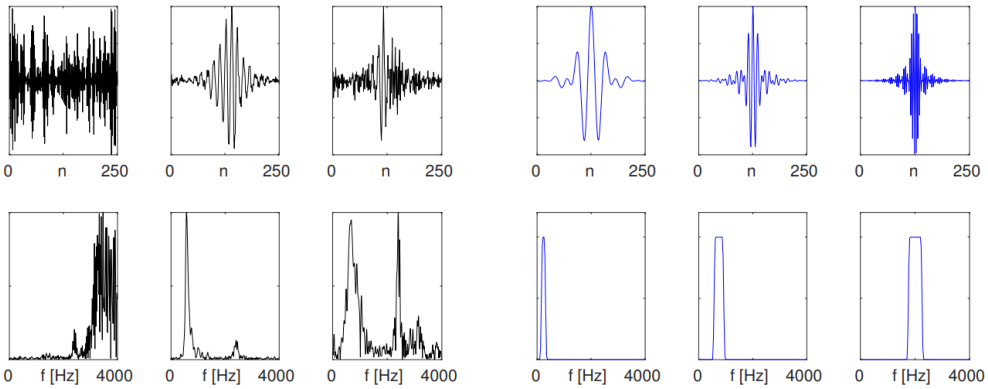


Fig3 : Représentation des filtres

Nous pouvons voir sur ces figures plusieurs filtres différents, à gauche proposé par un modèle classique et à droite par notre SincNet. En haut est leur représentation temporelle et en bas fréquentielle. Nous pouvons bien visualiser la simplicité des filtres du SincNet n'ayant pour paramètre que les fréquences de coupure, par rapport au traitement intégral du signal par un autre modèle. Par exemple, pour un signal composé de F filtres de longueur L, un modèle basique s'entraînera sur F*L paramètres et le SincNet sur 2*F.

X-Vector

Une fois que nous avons prétraité notre audio avec le SincNet, il est temps de réaliser notre embedding final. En conséquence nous utilisons le X-Vector[4] qui est modèle avec une architecture de réseaux neuronaux à délai temporel (TDNN), un type de modèle d'apprentissage profond (DNN) particulièrement adapté aux signaux sonores car il prend en compte les contextes locaux de son entrée.

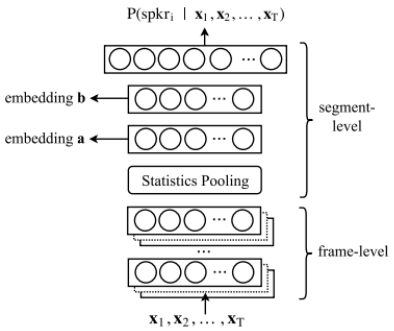


Fig4 : Diagramme du X-Vector

Comme nous pouvons le remarquer, notre vecteur de sortie du SincNet va passer par des couches d'apprentissage au niveau des frames (fenêtres locales) puis par une couche de Statistics Pooling afin de concentrer l'information et enfin des couches au niveau du segment en résultant, et nous pourrons récupérer notre embedding à l'avant-dernière couche avant la classification car nous voulons faire notre propre modèle.

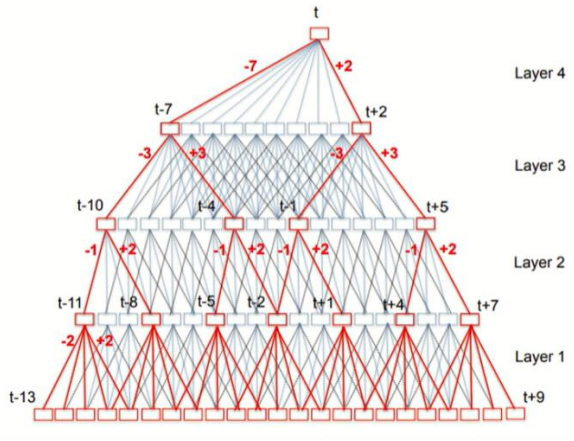


Fig5: Représentation des couches avant le Stats Pooling

Layer	Layer context	Total context	Input x output
frame1	$[t-2, t+2]$	5	120x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

Fig6: Tableau du X-Vector

Un modèle d'apprentissage profond classique aurait pris directement toutes les valeurs du vecteur d'entrée et aurait commencé l'apprentissage des neurones sur cet ensemble. Comme nous pouvons le voir sur la figure 5, nous créons des fenêtres locales de cinq valeurs de t-2 à t+2 compris, puis nous décalons d'une valeur pour la prochaine fenêtre et ainsi de suite. Nous répétons le même principe sur les couches supérieures afin de bien apprendre les caractéristiques du signal en prenant en compte le contexte du signal, car il y a bien une suite logique à ce que quelqu'un dit à l'oral, chaque mot sont corrélés et il nous faut un modèle prenant en compte ces corrélations au long terme. (C'est le même principe que ChatGPT utilise pour comprendre ce que vous dites et écrire correctement). Comme vous pouvez constater, cela améliore énormément notre modèle comparé à un réseau de neurones classique qui prendrait toutes les valeurs de l'entrée sans tenir en compte de leur position dans la séquence. Nous pouvons constater sur la figure 6, l'évolution du contexte total de chaque fenêtre à chaque couche ainsi que du nombre de valeurs en entrée et sortie de chaque couche. On se retrouve avant la couche de Stats Pooling avec des vecteurs de 1500 valeurs au nombre T, contexte total. La couche de Stats Pooling va justement créer un vecteur moyen ainsi qu'un vecteur écart-type à partir de tous ces vecteurs, puis les combiner. Et ensuite nous avons les couches au niveau du segment pour réduire l'embedding final à seulement 512 valeurs.

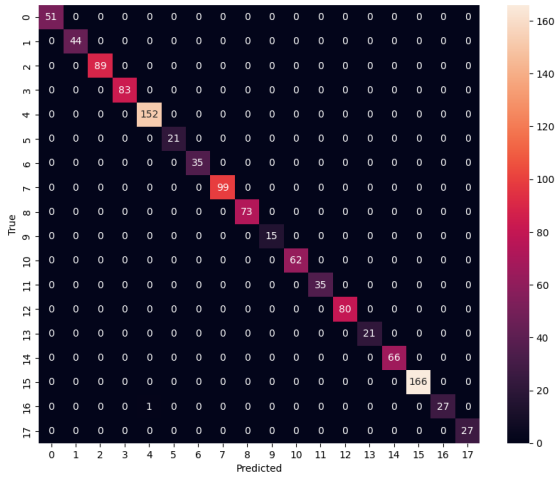
Classification par CNN

Une fois le vecteur sorti, il faut maintenant entraîner un modèle pour associer ces vecteurs d'audio avec leur locuteur. Pour cela, nous n'avons pas eu besoin d'aller chercher un modèle de classification pré-entrainé, nous avons simplement créée le nôtre. Pour ce qui est des données de type vecteur et

donc de type matrice comme les audios ou bien les images, les modèles d'apprentissage profond à convolution (CNN) sont les plus adaptés. Il s'agit de simples réseaux neuronaux avec des couches de convolutions pour extraire les caractéristiques de l'entrée (à l'image d'un filtre Sinc du SincNet qui sont en réalité un type de filtre de convolution). Plusieurs filtres, dans notre cas 32, sont choisis arbitrairement par le modèle et appliqués à notre vecteur pour encore une fois l'extraction de caractéristiques. Cela peut sembler redondant avec les modèles précédents mais nous ne voulions pas prendre de risques et obtenir les meilleurs résultats possibles. Des couches de MaxPooling sont également appliqués pour réduire les sorties des filtres à leurs informations les plus importantes. Enfin nous agrégeons ces résultats, puis entrainons la sortie avec une couche de neurones classique. Nous implémentons une couche de Dropout qui va désactiver aléatoirement des neurones lors des différentes phases de l'apprentissage afin d'éviter justement le surapprentissage sur nos données d'entraînement.

Résultats

Les résultats que nous obtenons après toutes ces opérations sont excellents, nous avons une précision de 99.91% sur notre jeu de données de test. Nous n'avons pas les chiffres exactes d'Alten mais d'après la déposante du projet il serait d'environ 85%, avec notamment des difficultés au niveau des prédictions confondant M. Macron et Mme Lepen.



Matrice de confusion du modèle sur le test set

Cette matrice de confusion représente les prédictions du modèle sur le test set par rapport à leur vraie identité. Nous pouvons voir qu'il n'y a qu'une seule erreur sur les 1147 audios, au niveau d'un audio au label 16 et prédit à 4. Cela correspond à un des audios de M. Bardella prédit comme étant de M. Mitterrand. Cela reste une erreur assez triviale car il n'existe aucune interaction entre ces deux personnes et n'en existera jamais.

Il faut bien noter que bien que ces résultats puissent paraître hors du commun, ils ne sont pas représentatifs pour un usage en cas réel. En effet les audios de très bonnes qualité comme ceux utilisés pour entrainer le modèle sont plutôt rare lors d'un débat politique. En réalité, il existe beaucoup d'interruptions de la parole et de voix chevauchantes, qu'il faut séparer avec notre modèle de séparation de voix. Bien que ce modèle fonctionne bien, les audios en résultant ne sont toujours

pas d'aussi bonne qualité que ceux de base. Nous n'avons pas eu assez de temps pour créer un véritable jeu de donnée d'audios séparé par notre modèle afin de tester l'efficacité du modèle sur ce type d'audio. Sur les 23 audios séparé sur lesquels nous avons testé le modèle, 19 ont finalement été correctement prédit, correspondant à une précision de 82.61% mais il faudrait le tester plus extensivement pour faire de véritables conclusions.

Pipeline

Les fonctionnalités ont été développées séparément par les différents membres de l'équipe puis ajouter une à une à une pour former la solution technique qui se présente sous la forme d'un pipeline écrite en Python qui permet à partir d'un fichier audio .wav de séparer les voix chevauchantes grâce à la segmentation de l'audio performée lors de la diarisation et de la détection des overlaps. Les audios extraits sont ensuite traités pour identifier les orateurs et transcris. Les transcriptions sont sauvegardées dans des fichiers texte.

La difficulté de réalisation de la solution technique réside dans la complexité individuelle des fonctionnalités qui reposent toutes sur des sujets pointus de recherche sur le domaine du traitement et l'analyse de la parole. La complexité des différentes tâches a été abordée dans les parties précédentes. Néanmoins, la solution présente une difficulté majeure qui n'a pas encore été traitée mais qui a déterminé le design de la solution technique. Lors de l'implémentation de la séparation des voix, les premiers résultats obtenus avec MossFormer 2 étaient mauvais. Une vingtaines d'audios complets ont été traités par le modèle qui échouait à chaque fois à séparer les voix des locuteurs. À la suite de l'échec, le travail de documentation a repris afin de comprendre l'erreur commise. Plongés dans les papiers scientifiques pendant plusieurs jours, l'équipe se rend compte de son incompréhension du modèle, en particulier la façon dont il a été entrainé. Les datasets d'entrainements des modèles de séparation de voix sont WSJ0-2mix et WSJ0-3mix. Ces datasets sont composés de conversations artificielles, ils sont composés uniquement de courts overlaps qui résultent de la superposition de deux locuteurs aléatoires. Ainsi, le modèle MossFormer 2 n'est capable de séparer les voix uniquement sur des audios contenant une très forte présence des voix chevauchantes ce qui ne simule pas du tout les conversations entendues lors d'un débat qui elles possèdent une minorité d'overlaps qui viennent séparer les très majoritaires tirades argumentatives des locuteurs. De ce constat, la nécessité de la segmentation de l'audio du débat qui était vue au début du projet comme optionnelle s'imposait. La détection d'overlap en utilisant la librairie pyannote permet d'extraire automatiquement les segments overlaps qui seront les seuls segments soumis à MossFormer 2 pour séparation des voix. Ce changement de design a considérablement aidé à l'amélioration des résultats de séparation des voix.

Un autre problème, dans la continuité de celui-ci, réside dans la détection des locuteurs pour les segments d'overlaps séparés. En effet, nous avons établi précédemment que, bien que nous ayons obtenu d'excellents résultats sur les données de test d'Alten, notre modèle est moins performant sur des enregistrements de moins bonne qualité, tels que ceux des overlaps séparés. Effectivement, avec le choix entre 18 politiciens différents, il est souvent plus difficile pour notre modèle de détecter le seul bon choix possible lorsque les données ne sont pas adaptées. Ainsi, à l'origine, sur les 23 enregistrements d'overlaps séparés, nous n'avions que 12 bonnes prédictions, soit une précision de 52,2 %. De plus, pour les mauvaises prédictions, il arrivait souvent que le politicien choisi ne soit même

pas présent dans le débat initial, ce qui rendait les résultats incohérents. Pour remédier à cela, nous avons préféré passer directement par le pipeline plutôt que par les modèles. En effet, nous avons décidé de faire nos prédictions sur les segments d'audio de monologues en premier (étant des enregistrements de meilleure qualité), pour ensuite bloquer les prédictions des segments d'audio séparés aux politiciens détectés lors des monologues. Grâce à cette approche, nous n'avons plus eu de prédictions incohérentes et avons pu correctement détecter 7 des enregistrements qui n'étaient pas bons précédemment. Ainsi, nous sommes passés de 12 à 19 bonnes prédictions sur 23 enregistrements, soit une précision de 82,6 %.

IV) Document d'architecture technique

Objectifs du système

Le système a pour objectif d'identifier les locuteurs d'un débat politique et de transcrire leur discours à partir d'un fichier audio de format WAV. Pour ce faire, le système doit passer par plusieurs étapes nécessaires :

- Segmentation de l'audio en performant la diarisation et la détection des overlaps (segments dans lesquels deux locuteurs parlent en même temps).
- Détection des segments monologues (segments dans lesquels un orateur parle seul sans être interrompu)
- Extraction des segments monologues de chaque locuteur et des segments overlaps
- Séparation des segments overlaps
- Identification des locuteurs
- Transcription de l'audio

Les critères de performances attendus sont la précision des modèles de diarisation, de détection des overlaps, de séparation de voix, d'identification des locuteurs et de transcription Speech-To-Text.

Principaux composants du système

Les principaux composants du système sont :

- Les pipelines pré entrainées de diarisation et de détection des overlaps de la librairie pyannote
- Le module de détection des monologues
- La pipeline pré entrainée du modèle de séparation de voix MossFormer 2
- Le modèle d'identification des locuteurs
- L' API de Google de transcription Speech-To-Text

Chaque composant permet d'accomplir une des étapes décrite dans la partie ci-dessus. Le modèle MossForrmer 2 adopte l'architecture encoder-decoder propre aux transformers. L'encoder encode le

signal d'entrée en extrait les features les plus importantes grâce à une convolution 1D. Le decoder performe l'opération inverse et reconstruit le signal de base en utilisant une convolution layer transposée. La séparation se fait grâce aux blocs MossFormer et Recurrent. Le module MossFormer utilise le principe de self-attention pour capturer les dépendances lointaines tandis que le module Recurrent capture les relations localisées entre les éléments de la séquence. Un réseau de masque qui utilise des filtres de convolution est utilisé pour séparer les signaux.

Le modèle d'identification est basé sur des fonctions Sinc, conçus spécifiquement pour s'adapter aux signaux audios, et qui implémentent des filtres passe-bande. Il permet d'analyser les différentes fréquences qui composent le signal sonore. La première couche du modèle SincNet que nous appliquons en entrée à notre signal sonore, permet par un apprentissage automatique de trouver les meilleurs filtres passe-bande à appliquer. Une série de filtre est utilisé pour extraire des caractéristiques plus précises en se concentrant seulement sur des bandes de fréquences spécifiques plutôt que l'audio entier.

Architecture du système

Les différents composants sont organisés pour former un pipeline qui prend en entrée un fichier audio de format WAV. Les pipelines pré entrainées de diarisation et de détection des overlaps segmente l'audio. Ces segments représentés par des tuples sont stockés dans des listes. Ces listes sont ensuite utilisées pour détecter les segments monologues. Les segments monologues et overlaps sont extraits et enregistrées en fichier audio dans leur dossier respectif. Les audios overlaps extraits sont séparés avec le modèle MossFormer 2. Les audios séparés sont stockés dans leur propre dossier et sont soumis au modèle d'identification qui crée un dictionnaire permettant d'associer chaque nom de fichier au nom de l'orateur. Les segments monologues sont coupés pour en sélectionner les 10 premières secondes puis sont enregistrés dans un nouveau dossier destiné à la prédiction des locuteurs. Ces audios coupés sont soumis au modèle d'identification permettant de créer un autre dictionnaire associant chaque label de la diarisation à l'identité de l'orateur. Enfin, les audios overlaps et monologues sont soumis à l'API de transcription, le résultat est stocké dans un fichier texte.

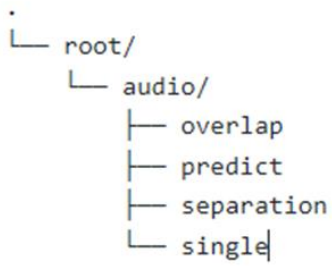


Figure 5 Arborescence du répertoire servant à stocker les différents audios

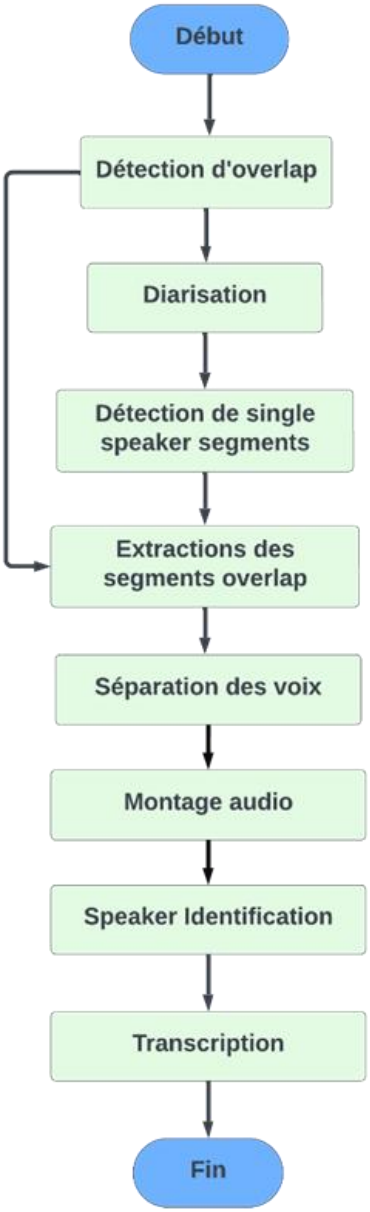


Figure 5 Arborescence de la pipeline du modèle

Technologies utilisées

Le pipeline est écrit exclusivement en Python. Les librairies utilisées sont :

- pyannote : pour la segmentation de l'audio
- pydub : montage/coupage des audios
- modelscope : accès à la pipeline pré entrainée du modèle de séparation de voix MossFormer 2
- SpeechRecognition : pour accéder à l'API Speech-To-Text de Google

Les librairies nécessaires sont regroupées dans un fichier *requirements.txt*.

Principaux risques et contraintes

La complexité du domaine du projet entraine un certain nombre de contraintes. D'un point de vue technique, la complexité des données audios de conversations humaines nécessite des modèles complexes de deep learning pour être analysées. Cela représente une dette technique importante. La séparation de voix est un domaine très actif de recherche qui présente des avancées de plus en plus techniques. Cependant, malgré des résultats satisfaisants pour un grand nombre d'extraits analysés, la fiabilité de ces modèles n'est pas encore parfaite ce qui compromet l'ensemble de la performance du pipeline qui dépend lourdement du résultat de séparation des overlaps. Les modèles utilisés nécessitent d'être entrainé sur de larges datasets qui demande une grande puissance calculatoire. Cela constitue un coût financier important en utilisation d'infrastructure cloud auquel s'ajoute l'utilisation d'une API Speech-To-Text privée comme celle de Google.

Comme dans toutes utilisations de modèles de machine learning, le risque éthique existe. Les systèmes d'analyse de discours politiques peuvent présenter involontairement des biais en fonction de facteurs tels que l'identité et l'idéologie politique. Aborder les biais et garantir l'équité dans la collecte de données, l'entrainement des modèles et les méthodologies d'analyse est essentiel pour prévenir les implications éthiques.

Déploiement et la maintenance du système

Le déploiement et la maintenance du système exigent une approche méthodique et continue. Tout d'abord, il est essentiel d'établir une infrastructure robuste, comprenant des serveurs, des bases de données qui peut soutenir les exigences de traitement et de stockage des données audio. Une fois le système déployé, une surveillance constante est nécessaire pour monitorer les performances, détecter les erreurs et résoudre les problèmes éventuels. A cause des phénomènes de concept drift et de data drift, des mises à jour régulières des modèles d'analyse et des algorithmes sont nécessaires pour améliorer la précision et l'efficacité du système au fil du temps. Enfin, c'est avant tout un projet de recherche et développement, essayer d'implémenter d'autres approches à la résolution du problème est crucial pour atteindre des performances de l'état de l'art.

V) Retrospective

À travers ce projet d'analyse politique automatisée, notre équipe a franchi des étapes significatives et jeté les bases pour des développements futurs. Cette phase initiale a permis de poser les fondements d'un système prometteur, tout en identifiant des axes d'amélioration et des perspectives d'évolution.

Difficultés et Défis, un projet de recherche :

Ce projet s'est avéré être à la pointe de la technologie, obligeant notre équipe à travailler sur un projet complexe qui se rapproche davantage d'un travail de Recherche et Développement (R&D). En effet, la nature même de l'analyse automatique des discours politiques à partir de fichiers audios représente un défi technologique majeur. Nous avons dû constamment rester à l'affût des dernières avancées technologiques, effectuer une veille constante sur les nouveaux développements dans le domaine du traitement de la parole et de l'intelligence artificielle, ainsi qu'étudier attentivement les papiers de recherche les plus récents. Cette exigence de rester à la pointe de la technologie a été une caractéristique unique de ce projet, le distinguant nettement des autres projets que nous avons réalisés à l'EFREI.

Réussites de la collaboration en équipe :

Travailler en équipe de cinq personnes a été une réussite majeure de ce projet. Malgré sa taille conséquente par rapport à nos autres projets, notre équipe a su tirer parti de cette diversité pour répartir efficacement les tâches et travailler en parallèle. Répartir le travail en plusieurs piliers (diarisation, identification, voix superposées) majeurs nous a permis de rester efficace sans dépendre du travail des autres membres de l'équipe. Cette approche nous a permis de répondre à une contrainte de temps restreinte en réalisant des progrès significatifs sur plusieurs fronts simultanément. En assignant des tâches réalisables en parallèle, nous avons pu maximiser notre productivité et avancer efficacement vers nos objectifs communs. Cette capacité à collaborer harmonieusement en équipe, malgré la complexité du projet et la contrainte de temps, a été un élément clé de notre succès.

Bilan et Retombées Actuelles

Nous avons réussi à ériger une infrastructure technique solide, accompagnée du développement d'algorithmes avancés pour la diarisation de locuteur, la transcription audio, et l'identification des intervenants. Ces réalisations constituent des jalons majeurs dans la concrétisation de notre objectif d'analyse politique automatisée.

Par ailleurs, l'adoption d'une méthodologie agile s'est avérée fructueuse, permettant une gestion efficace du projet et une collaboration harmonieuse entre les membres de l'équipe. Cette approche nous a dotés d'une agilité nécessaire pour répondre aux exigences changeantes et maintenir un haut niveau de qualité tout au long du processus de développement.

Perspectives d'Amélioration et d'Évolution

Pour consolider notre système d'analyse politique automatisée, plusieurs axes d'amélioration et de développement émergent, conformément aux défis rencontrés et aux opportunités.

Affinement des Modèles dans les Pipelines : L'optimisation des modèles utilisés dans nos pipelines constitue une priorité. Nous nous engageons à affiner ces modèles afin de les rendre plus adaptés aux spécificités des débats politiques, visant ainsi à améliorer la performance globale de notre système.

Constitution d'un Nouveau Dataset de Débats : La création d'un nouveau dataset, comportant des débats avec des chevauchements, représente un impératif. Nous prévoyons d'annoter ce corpus de données afin d'enrichir nos modèles et d'améliorer leur capacité à gérer des situations plus complexes.

Définition de Métriques Objectives : Pour évaluer l'efficacité de nos modèles de manière plus précise, il est essentiel de définir des métriques objectives. Ces mesures nous permettront d'évaluer la performance de notre système de manière quantitative, dépassant ainsi les évaluations manuelles souvent sujettes à la subjectivité.

Poursuite du Développement : Nous envisageons également de poursuivre le développement de notre système en explorant de nouvelles fonctionnalités telles que la détection de thèmes, l'analyse d'émotions, le résumé de texte, et l'analyse du temps de parole. Ces extensions enrichiront notre capacité à extraire des insights pertinents à partir des discours politiques.

En somme, ces perspectives d'amélioration témoignent de notre engagement continu à perfectionner notre système d'analyse politique automatisée, afin de fournir des résultats toujours plus précis et significatifs dans la compréhension des discours politiques contemporains.

Perspectives d'Application et d'Impact :

Les implications de notre projet sont vastes et diverses. Notre système pourrait offrir des insights précieux aux analystes politiques, aux journalistes et aux décideurs, éclairant leurs prises de décision et leur permettant d'appréhender plus finement les nuances des discours politiques.

En outre, l'intégration de notre technologie dans des plateformes de médias sociaux ou de diffusion en direct pourrait révolutionner la manière dont les discussions politiques sont perçues et participées, favorisant une démocratie plus informée et participative.

En conclusion, notre projet représente une étape significative vers l'automatisation de l'analyse politique, ouvrant la voie à de nouvelles avenues d'exploration et de compréhension des discours politiques contemporains. Nous sommes résolument engagés à poursuivre nos efforts et à explorer les nombreuses opportunités qui se présentent à nous dans ce domaine en constante évolution.

Conclusion

Ce projet a constitué une opportunité précieuse pour l'acquisition et le renforcement de compétences variées, couvrant à la fois les domaines techniques et interpersonnels. L'implication dans le projet Méd-IA a non seulement permis d'élargir nos connaissances en matière de traitement audio et de recherche et développement, mais a également souligné l'importance de l'innovation dans les applications démocratiques.

En participant à ce projet, nous avons acquis une expérience significative, nous confrontant à des défis uniques tout en favorisant notre développement professionnel. Ce projet se distingue par son originalité et son approche de recherche avancée, marquant une contribution notable à notre domaine d'étude.

Nous sommes très fières d’y avoir participé.

Annexes

Mots clés :

Diarisation : La diarisation du locuteur est le processus de partitionnement d'un flux audio contenant de la parole humaine en segments homogènes en fonction de l'identité de chaque locuteur (Wikipédia).

Embedding : Une représentation numérique d'un objet (dans ce cas, un fichier audio) dans un espace vectoriel. Cette représentation est créée de manière à capturer les caractéristiques importantes de l'objet pour une tâche spécifique, comme la classification des locuteurs.

Pipeline : Une pipeline est une séquence d'étapes interconnectées de traitement des données et de modélisation. Son objectif est d'automatiser, de standardiser et de simplifier le processus de construction, de formation, d'évaluation et de déploiement des modèles d'apprentissage automatique.

Sources :

Diarisation :

GitHub de Pyannote publié par Hervé Bredin https://github.com/pyannote

Pyannote vs Nemo par La Javaness R&D sur Medium https://lajavaness.medium.com/comparing-state-of-the-art-speaker-diarization-frameworks-pyannote-vs-nemo-31a191c6300

Séparation des voix :

MossFormer2: Combining Transformer and RNN-Free Recurrent Network for Enhanced Time-Domain Monaural Speech Separation by Shengkui Zhao, Yukun Ma, Chongjia Ni, Chong Zhang, Hao Wang, Trung Hieu Nguyen, Kun Zhou, Jia Qi Yip, Dianwen Ng, Bin Mahttps://arxiv.org/pdf/2312.11825.pdf

Identification de locuteur:

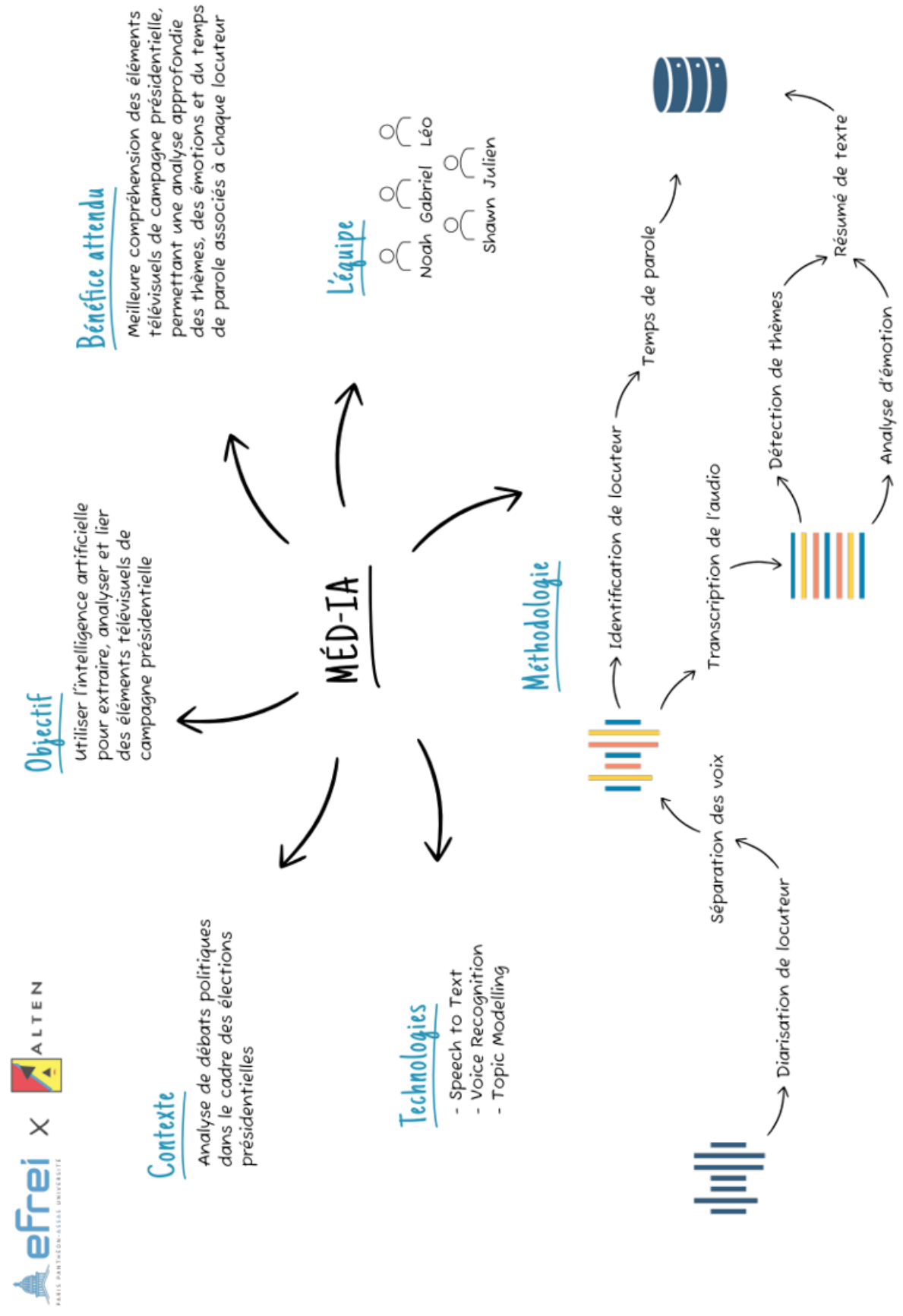
Human Voice Recognition Depends on Language Ability by Tyler K. Perrachione, Stephanie N. Del Tufo, John D.E. Gabrieli

wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations by Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli

Speaker Recognition from Raw Waveform with SincNet, by Mirco Ravanelli, Yoshua Bengio

Deep Neural Network Embeddings for Text-Independent Speaker Verification, by David Snyder, Daniel Garcia-Romero, Daniel Povey, Sanjeev Khudanpur

Sketchnote :


MÉD-IA
Objectif
Utiliser l'intelligence artificielle pour extraire, analyser et lier des éléments télévisuels de campagne présidentielle
Bénéfice attendu
Meilleure compréhension des éléments télévisuels de campagne présidentielle, permettant une analyse approfondie des thèmes, des émotions et du temps de parole associés à chaque locuteur
L'équipe
Noah Gabriel Léo
Shawn Julien
Méthodologie
Identification de locuteur
Temps de parole
Transcription de l'audio
Détection de thèmes
Résumé de texte
Analyse d'émotion
Séparation des voix
Diarisation de locuteur
Technologies
- Speech to Text
- Voice Recognition
- Topic Modelling
Contexte
Analyse de débats politiques dans le cadre des élections présidentielles
eFrei X ALTEN

Poster :

MÉD-IA

Léo AVRIL
Shawn DUHAMEL
Gabriel GEORGE
Noah GOULIN
Julien VIEILLEVIGNE

OBJECTIFS

- Analyser les vidéos de campagne présidentielle en utilisant l'intelligence artificielle.
- Obtenir une compréhension approfondie des thèmes abordés dans les discours politiques.
- Identifier et évaluer les émotions exprimées par les candidats et les électeurs dans les vidéos de campagne.
- Quantifier et comparer le temps de parole de chaque candidat pour une analyse plus équilibrée et approfondie.

CONTEXTE

Dans le contexte des élections présidentielles en France, les analyses reposent largement sur les sondages et les résultats électoraux, négligeant souvent les analyses automatisées des débats politiques, pourtant cruciales en période de campagne.

METHODOLOGIE

- Détection de locuteur
- Séparation de voix
- Identification de locuteur
- Transcription de l'audio
- Mise en pipeline

RESULTATS

- Réalisation d'une pipeline réunissant diarisation, détection de chevauchements et identification de locuteur
- >99% de précision pour l'identification de locuteur
- Transcription labélisée d'audios de débat à l'écrit