جامعة سيدي محمد بن عبدالله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES
المدرسة العليا للتكنولوجيا
ECOLE SUPÉRIEURE DE TECHNOLOGIE

UNIVÉRSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

ÉCOLE SUPÉRIEURE DE TECHNOLOGIE

DÉPARTEMENT INFORMATIQUE
FILLIÈRE GÉNIE INFORMATIQUE

APPLICATION DE DÉTECTION DE LANGUE ET RÈSUMÈ AUTOMATIQUE EN BASE DE NLP & MACHINE LEARNING

Réalisé par

Bendakhkhou HIBA

Encadré par

MOHAMMED Benslimane

Remerciement

Je tiens à travers ce rapport a exprimé mes sincré remerciement a tout l'équipe pédagogique de ESTF , Je tiens à exprimer ma profonde reconnaissance envers les professionnels responsables de la filière Génie informatique pour m'avoir offert une formation de qualité.

Je souhaite également exprimer ma gratitude envers mon encadrant de PFE, le professeur **M. BENSLIMANE Mohammed**, pour ses précieux conseils, sa disponibilité et son soutien tout au long de ce projet, et qui n'a ménagé aucun effort pour m'aider et me soutenir, ainsi que pour ses conseils précieux et ses directives pertinentes.

Je tiens aussi à remercier les membres du jury qui ont pris le temps de participer à la soutenance et d'évaluer mon travail. Enfin, je souhaite exprimer ma gratitude à tous ceux qui ont contribué, de près ou de loin, à la réussite de ce travail. Un grand merci du fond du cœur.

Table des matières

I. Cahier des charges

Introduction

Ces dernières années ont vu une montée en importance de deux sous-domaines fondamentaux de l'intelligence artificielle : l'apprentissage automatique et le traitement automatique du langage naturel (NLP).L'apprentissage automatique révolutionne la manière dont les systèmes informatiques peuvent apprendre à partir de données et améliorer leurs performances au fil du temps sans programmation explicite. Grâce à l'utilisation de modèles statistiques et d'algorithmes, l'apprentissage automatique permet aux ordinateurs d'identifier des schémas complexes dans les données, leur permettant de prendre des décisions autonomes et d'accomplir efficacement des tâches spécifiques.

Au même sens, le traitement automatique du langage, ou NLP, se concentre sur la manière dont les machines peuvent comprendre et interagir naturellement avec le langage humain. Des techniques avancées telles que l'analyse syntaxique, la reconnaissance vocale et la traduction automatique permettent au traitement automatique du langage naturel (NLP) de permettre aux ordinateurs de comprendre et de produire du langage humain dans une variété de contextes. Cela ouvre la porte à une large gamme d'applications, notamment les chatbots, les systèmes de traduction automatique et les systèmes de recommandation de contenu.

En combinant le NLP & le machine_learning, nous créons une collaboration puissante qui permet aux machines de comprendre, d'interpréter et de produire du langage naturel de manière plus avancée,tel-que l'amélioration de la capacité des machines à traiter saisir le sens et le contexte du langage humain, ainsi qu'à générer des réponses pertinentes et naturelles. Cela ouvre de nouvelles perspectives passionnantes pour l'avenir de l'intelligence artificielle et de ses applications dans divers domaines.

1. Présentation du projet

Avec la multiplication des contenus disponibles sur Internet, il est devenu essentiel de comprendre et d'analyser rapidement les informations textuelles. Dans un environnement en constante évolution, mon projet s'est concentré sur le développement d'une solution complète et efficace, exploitant les techniques avancées de traitement automatique du langage naturel (NLP).

L'objectif principal de mon projet était de fusionner les technologies de NLP et d'apprentissage automatique pour créer une application capable d'identifier automatiquement la langue d'un contenu audio, puis de le transcrire en texte pour une analyse ultérieur afin de donné un résumé complet., et cela avec une intégration des algorithmes de NLP et des modèles d'apprentissage automatique pour accomplir cette tâche et découvrir des caractéristiques avancées permettant de reconnaître et de traiter les différentes langues parlées dans les fichiers audio, puis de les transcrire avec une grande précision.

Une fois le contenu audio converti en texte, l'application doit produire un résumé concis et informatif du contenu extrait, offrant ainsi aux utilisateurs une vue d'ensemble rapide et pertinente des informations contenues dans le fichier audio.

2. Objectifs du projet

L'objectif principal de mon projet était de fusionner les technologies de NLP et d'apprentissage automatique pour créer une application capable d'identifier automatiquement la langue d'un contenu audio, puis de le transcrire en texte pour une analyse ultérieur afin de donné un résumé complet.

il peut s'apparaitre que le projet vise a réalisé deux taches distinct ,mais il sont également interconnectée ,par ce que en tant que utilisateur international,qui cherche d'un résumé afin d'obtenir une information exact important , d'aprés

un document qui peut être dans une langue incompréhensive pour lui, c’est pour cela ,La détection de langue permet d'obtenir une expérience personnalisée en recevant du contenu dans leur langue préférée et facilite la classification et l'organisation des données dans des environnements multilingues.

La Fonctionnalité de detection de langue revêt une importance capitale dans divers contextes, notamment pour les services de traduction automatique, les moteurs de recherche multilingues et les plateformes de médias sociaux. La détection de langue permet aux utilisateurs de bénéficier d'une expérience personnalisée en recevant du contenu dans leur langue préférée, et facilite également la classification et l'organisation des données dans des environnements multilingues.

D’autre part , La fonctionnalité de résumé automatique est particulièrement utile dans les cas où les utilisateurs ont besoin d'obtenir rapidement une compréhension générale du contenu sans avoir à lire ou écouter l'intégralité du texte ou de l'audio.

Cette combinaison présente de nombreux bénéfices, tels qu'une expérience utilisateur optimisée, une productivité renforcée et une compréhension améliorée du contenu dans un contexte multilingue.

Voici quelque avantage :

Personnalisation et Accessibilité : L'application peut détecter automatiquement la langue dans laquelle le contenu est rédigé et générer un résumé dans cette langue, ce qui offre une expérience personnalisée aux utilisateurs et facilite l'accès à l'information dans leur langue préférée.

Gain de Temps : En générant automatiquement des résumés, l'application permet aux utilisateurs de gagner du temps en évitant de devoir parcourir l'intégralité du contenu pour en extraire les informations pertinentes.

Analyse Multilingue : L'application peut être utilisée pour analyser et résumer des contenus provenant de différentes langues, ce qui facilite la comparaison et l'analyse de données dans un contexte multilingue.

2.1. À qui s'adresse ce projet

le projet s'adresse à diverses parties prenantes et peut bénéficier à différents utilisateurs,il peut être :

Utilisateurs Individuels : Les personnes qui souhaitent obtenir un résumé rapide d'un contenu textuelle dans une langue spécifique sans avoir lire tout sont contenu. Cela peut inclure des étudiants, des professionnels pressés ou toute personne cherchant à gagner du temps lors de la consommation de contenu audio.

Entreprises de Médias et de Veille : Les entreprises qui traitent de grandes quantités de contenu et qui ont besoin de méthodes efficaces pour extraire les informations essentielles. Alors,le projet peut leur fournir des résumés automatiques de manière rapide et efficace, ce qui facilite leur analyse et leur prise de décision

Services de Traduction et d'Interprétation : Les services de traduction automatique et d'interprétation peuvent l'utilisé pour détecter automatiquement la langue dans laquelle un contenu audio est rédigé, ce qui peut faciliter le processus de traduction en identifiant rapidement la langue cible.

Équipes de Recherche et d'Analyse : Les équipes de recherche et d'analyse qui travaillent sur des projets multilingues peuvent utiliser ce projet afin d'analyser et résumer efficacement des contenus audio dans différentes langues, ce qui leur permet d'obtenir rapidement des informations pertinentes.

Développeurs et Innovateurs : Les développeurs intéressés par le domaine du traitement automatique du langage naturel et de l'apprentissage automatique peuvent utiliser notre projet comme une base pour développer de nouvelles applications et solutions dans ce domaine.

Le projet offre une solution polyvalente et efficace pour répondre aux besoins diversifiés d'utilisateurs et d'organisations. En combinant la détection de langue, la conversion audio en texte et la génération de résumés automatiques, notre application permet d'optimiser la gestion et l'analyse de contenu multilingue. Cela offre une valeur ajoutée significative dans des domaines variés tels que la recherche d'information, l'analyse de contenu et bien d'autres encore.

2.2. Etat actuel des ressources et fonctionnement du préexistant

Avant de commencer dans le processus de développement de projet, il faut premièrement évaluer l'état actuel des ressources disponibles ainsi que le fonctionnement du système préexistant, le cas échéant. Cette évaluation nous permettra de comprendre les fondations sur lesquelles le projet sera basé dans son réalisation , et de déterminer les défis potentiels à rencontrer en cours de route.

Dans cette section, il y a une examination ressources disponibles, telles que les données d'entraînement, et les technologies disponibles.

Les resources actuelles :

Données d'Entraînement :

Le projet est divisé en trois parties distinctes. La première concerne la détection de langue d'un contenu audio ou textuel, la seconde se concentre sur le résumé automatique, tandis que la troisième partie porte sur la génération de résumés de texte. En termes de données d'entraînement, la partie audio nécessite une quantité significative de données, suivie de la partie texte.

Cependant, la partie de résumé automatique ne nécessite pas spécifiquement de données d'entraînement.

Après des recherches approfondies, voici les résultats obtenus :

Pour la partie audio, qui requiert une importante quantité de données, j'ai découvert une base de données contenant des enregistrements audio dans trois langues spécifiques : le français, l'allemand et l'anglais. Cette base de données est la plus conséquente que j'ai trouvée dans la communauté Kaggle, avec une taille de 16 Go.

pour la partie text (detection de langue), sur le même site, j'ai trouvé un fichier CSV contenant un ensemble de phrases dans plusieurs langues, chaque phrase étant associée à sa langue respective. Les langues couvertes comprennent l'arabe, le français, l'anglais, l'allemand, le russe, l'hindi, entre autres. La taille totale du fichier est de 1 911 016 octets.

Technologies Disponibles :

la partie audio :

Tout d'abord, la transformation de l'audio en spectrogramme Mel-Fréquence Cepstral Coefficients (MFCC). Ensuite, l'application d'un réseau neuronal (NN) afin de déterminer les caractéristiques fréquentielles de chaque langue. La réalisation de cela nécessite la présence des bibliothèques suivantes :

NumPy et **SciPy** pour le traitement des données et les calculs scientifiques :

Ils sont deux bibliothèques essentielle pour le traitement des données qui offrent des structures de données puissantes pour manipuler des tableaux multidimensionnels ainsi que des fonctions pour effectuer des opérations

mathématiques sur ces tableaux,il serons sera utilisé pour représenter les données audio sous forme de tableaux et effectuer des opérations de transformation et de traitement nécessaires pour obtenir les MFCC spectrogram

Matplotlib pour la visualisation des spectrogrammes :

Cette representation visuel sera nous permet d'analyser les caractéristiques fréquentielles par lesquelle nous serons capable d'identifier la langue de contenue audio.

TensorFlow avec **Keras** pour la création du réseau neuronal (NN) :

Ces deux bibliothèques Ensemble, offrent un environnement puissant pour la création, l'entraînement et le déploiement de modèles de ML. Et ils seront nous permet de créer et entraîner un réseau neuronal qui analyse les caractéristiques fréquentielles des spectrogrammes.

La partie text :

Pour traiter le contenu textuel, il faut d'abord le préparer avant d'appliquer le modèle de machine learning qui nous permettra de détecter la langue. Cette préparation implique l'application de techniques de traitement du langage naturel (NLP), suivie de la visualisation de la base de données. Pour ce faire :

Pandas : Pandas offre des structures de données flexibles et des outils pour manipuler et analyser des données. Nous l'utiliserons pour visualiser la base de données textuelles, notamment pour explorer ses caractéristiques et sa distribution.

NLTK (Natural Language Toolkit) : Nous l'utiliserons pour effectuer les traitements NLP nécessaires, tels que le nettoyage du texte, la tokenization, le stemming ou le lemmatization. Ces étapes sont essentielles pour préparer les données textuelles avant de les utiliser dans le modèle de détection de langue.

Math : La bibliothèque Math est une bibliothèque standard de Python qui fournit des fonctions mathématiques de base. Elle peut être utilisée pour certaines opérations mathématiques nécessaires dans l'implémentation du modèle choisi.

Pour le résumé automatique*:*

Dans ce cas, en utilisant uniquement NLTK et la bibliothèque mathématique de Python,puique nous serons implémenter une méthode de résumé automatique qui se concentre sur le traitement du langage naturel (NLP) et appliquer les techniques mathématiques pour générer des résumés.

NLTK (Natural Language Toolkit) : pour effectuer les traitements de base du texte, tels que la tokenization, la lemmatization, la suppression des mots vides (stopwords), et éventuellement l'analyse de la syntaxe pour identifier les phrases clés ou les relations entre les mots. Ces étapes permettront de préparer le texte avant la génération du résumé.

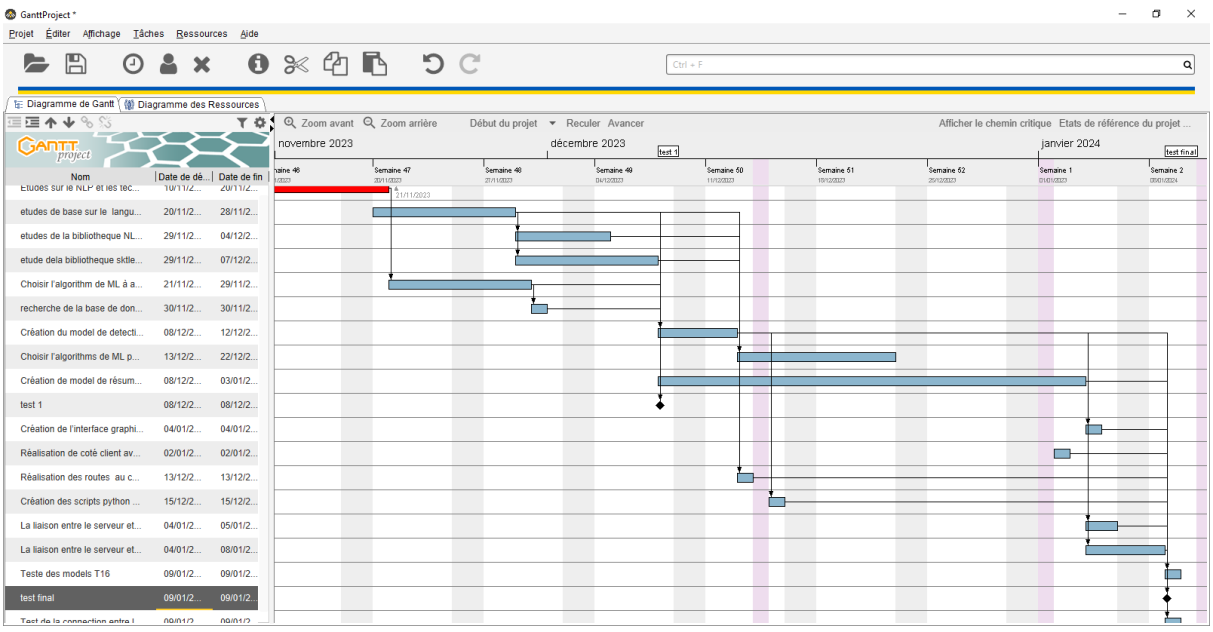
Bibliothèque mathématique (Math) : pour les techniques mathématiques qui sert à évaluer l'importance des phrases ou des mots dans le texte

Pour l'interface graphic :

Pour créer une interface utilisateur, il faut d'abord mettre en place des routes dans un serveur Node.js. Ensuite, nous utilisons AJAX pour communiquer avec le serveur côté client. Pour intégrer notre modèle Python, nous devons établir une communication entre le serveur Node.js et notre script Python. Dans mon cas, j'ai utilisé le module `child_process`, qui est intégré à Node.js et permet de créer des processus enfants. Ces processus enfants sont des processus indépendants du processus principal de Node.js et peuvent exécuter des commandes dans le shell du système d'exploitation. Nous utilisons spécifiquement la méthode `spawn` pour envoyer les paramètres nécessaires à l'exécution du modèle et récupérer le résultat pour l'afficher dans l'interface graphique. Pour la partie front-end, nous utilisons HTML et CSS pour créer une interface statique.

2.3. conception

Image représente le diagramme de gantt de projet :



3. Presentation attendues

3.1 Création,réalisation et récupération de contenue

La création, la réalisation et la récupération de contenu représentent des étapes fondamentales pour alimenter notre système de détection de langue et de résumé automatique. Dans cette section, nous détaillons les processus impliqués dans la collecte, la création ou l'acquisition de données textuelles et audio pour notre application.

Collecte de Données :

Comme déjà mentioné dans la partie 1.4 ,on a obtenu les données de training et de testing d'après la plateform kaggle,au coté audio c'est une base de données (audio de format .wav)qui contient 16GB des audio en trois langue l'allmand ,le francais et l'anglais, pour chaque languages ,mal et femal, et pour le coté textuelle ,on a utilisé une base de donnes qui contienty des phrase en different langues tel qui AR,FR,E,EN,SP..),chaque ligne deux colones une de langues et l'autre de la phrase.

Création de contenue :

Les etapes naicessaires afin de créer notre projet sont les suivant :

Partie1 : **detection de langue d'un audio**

Dans cette partie on doit premierement preparer les donnes afin d'appliquer un NN sur elle,la preparation des donnees est l'equivalence de la transformation des audio en MFCC spectrogram

Définition :
Un spectrogramme est la représentation visuelle d'un son,qui est basé sur le changment de la fréquence par le temp.

pourquoi les spectrogram ?

les signal contient plusieurs composant fréquencielle ,qui change 'une personne a une autre,et 'une launge a une autre ,car en peut trouve que la tonalité 'une pérsonne se change entre eux langue different,alors c'est la meilleurs notion qu'on peuve utilisé et etuier a fin de clasiffier les audio. Et pour mieux avoir les caractéristique fréquentielle de l'auio,le spectrogram rest le meilleur choix ,
les étapes de réalisation de spéctrogram :

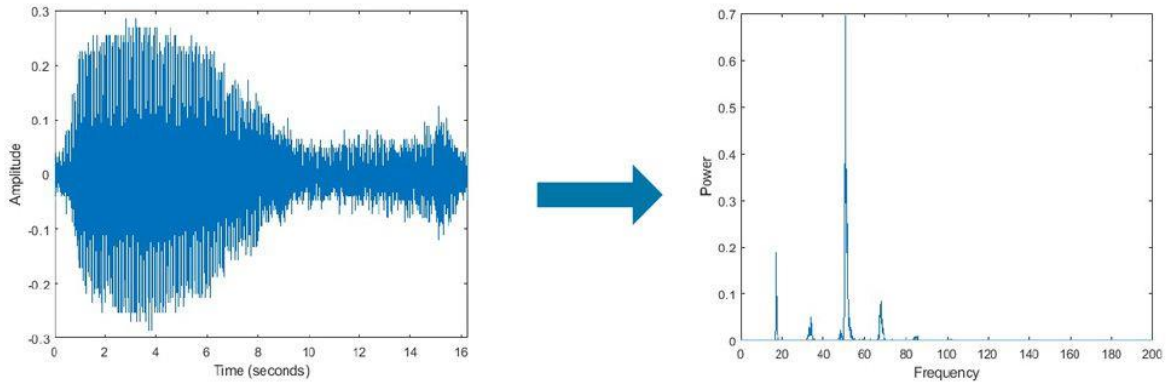
ans cette partie nous serons voirs 3 aspects,pratique ,mathematique et aspect traitement (code) avec python

les format es signal audio son nombreu,.WAV,.FLAC,MP3,chacun utilisé dépond au besoin :
.**MP3** : format audio compressé qui réduit la taille des fichiers audio en éliminant certaines données auditives moins essentielles . Il est utilisé pour la diffusion de musique en ligne et le stockage de fichiers audio sur des appareils portables puisqu'il est d'une taille réduite.
.**FLAC** : un format audio sans perte, qui comprime les données audio sans perte de qualité et utilisé par les audiophiles pour archiver leur musique dans une qualité audio haute fidélité.
.**WAV** : le format standard pour les fichiers audio sur les ordinateurs Windows. Les données audio son non compressées, et offrent une qualité audio élevée, mais occupent plus d'espace sur le disque dur.

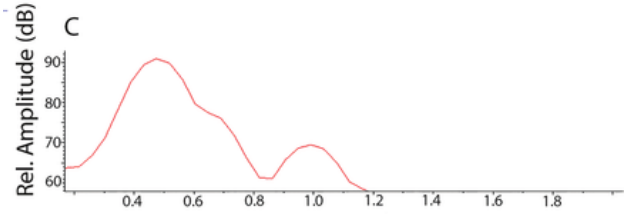
- a fin d'étudier touts les caractéristique d'un signal audio,on oive utilisé une format qui sert a garder les fonctionalité de l'audio, c-à-d .WAV ou .FLAC ,ans mon cas 'ai obtenu un ensembles des données qui son es audios en trois langue (GERMAN,ENGLISH,FRENSH),dans chaque langue il y a une partie femal et une partie mal,c'est e 16GB ,ans laquelle les auio son en format .FLAC.

Les étapes :

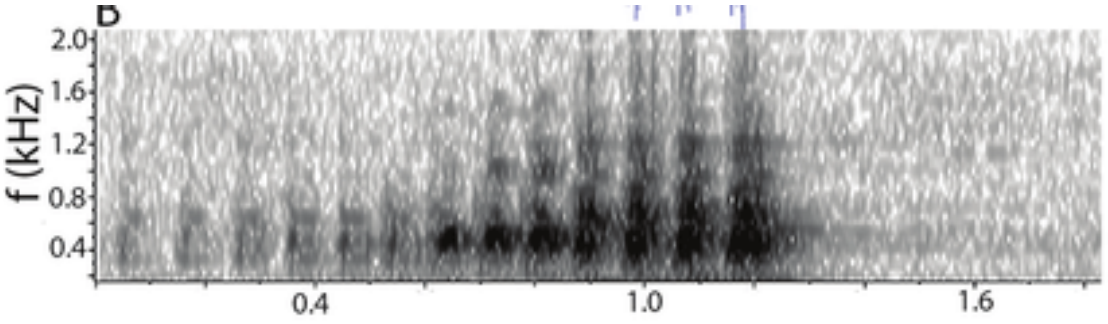
1.le fenêtrage : premierement en commane a découper notre signal en plusieurs petit fenêtre, chaque fenêtre a une taille spécifique, et la duréé entre deux fram succesive est nommmé hope size ,l'ors de fenêtrage,on utilise la methoe de overloading,cad la deuxième fenêtre est inclue dans la premiêre, autrement dit ,le hpt size et inférieur de la taille de la trame,on utilise cette méthode pour ne pas pérdue aucune partie e notre signal audio.
2.FFT :
Le mot FFT est une abréviation e (Fast Fourier Transform),il nous permet de passer e la representation de signal en fonction de temp vers sa representation en fanction e la fréquence,c'est le même concept du spectrogram ,pour une fenêtre uniquement.



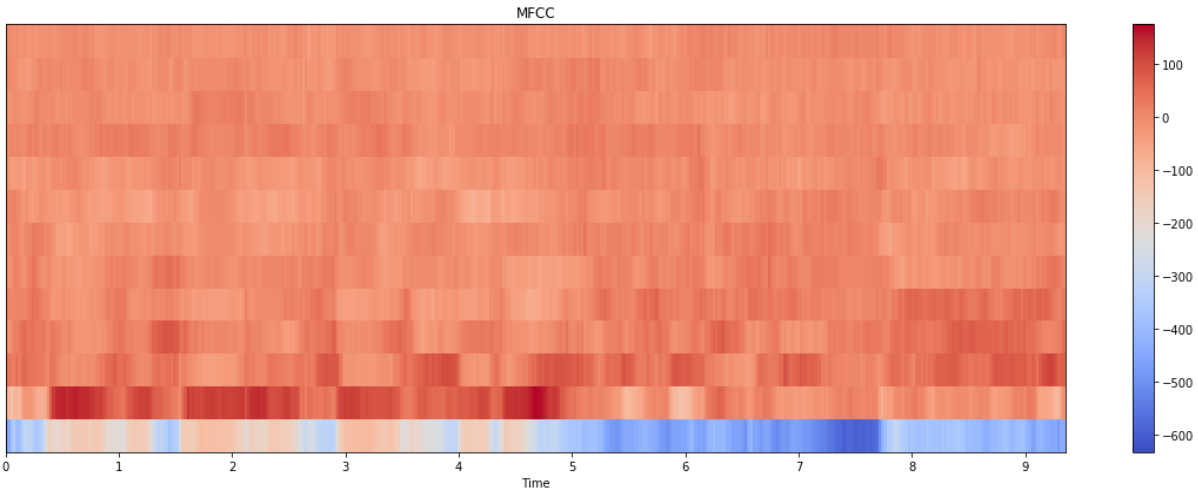
Afin d'obtenir le FFT d'une fenêtre,on doit premierement créé un spectrom,le spectrom c'est comme un representation des changement de l'amplitude de signal en fonction e leur fréquence



puis en faire une rotation par 90° e ce dernière (ce n'est pas d'une rotation mais comme l'inverse de la representation précédant),après cette étaps ,nous commence a attribuer pour chaque fréquence,une couleur spécifier,on peut utilisé soit gray scal [0,255] càd 256 possibilité,ou RGB[0,255]a la puissance trois (pour le rouge,le vers,le bleu, cà 256*256*256,la fréquence élevé est representé par une couleurs foncé.
Avec grey scal :



Avec RGB :



Après qu'on créé notre fft pour chaque fenêtre,en combine toutes ces FFT ,et en obtenir a la fin notre spectrogram.

Le MFCC vs Mel Spectrogram :

1.mel Spectrogram :

Def : Un Mel spectrogram est une représentation visuelle du spectre des fréquences dans un signal audio au fil du temps. Il affiche l'intensité des différentes composantes de fréquence dans le signal audio, utilisant généralement une palette de couleurs pour représenter l'intensité. Il fournit une représentation en 2D des données audio, où l'axe des x représente le temps, l'axe des y représente la fréquence (généralement sur l'échelle de Mel), et l'intensité de couleur représente l'amplitude ou l'énergie de chaque composante de fréquence à différents intervalles de temps.

- MFCC

Def : Les coefficients cepstraux en fréquence de Mel (MFCC, pour Mel-Frequency Cepstral Coefficients) sont un ensemble de coefficients qui capturent les caractéristiques spectrales d'un signal audio. Ils sont dérivés du spectrogramme de Mel mais sont ensuite traités davantage pour extraire des informations pertinentes. Les MFCC sont généralement utilisés comme caractéristiques pour des tâches telles que l'analyse de la parole et du son. Ils capturent des informations sur la forme de l'enveloppe spectrale, ce qui peut être utile pour des tâches telles que la reconnaissance vocale.

Bien que les spectrogrammes de Mel et les MFCC soient dérivés de la même échelle de Mel, ils servent des objectifs différents. Le spectrogramme de Mel fournit une représentation temps-fréquence du signal audio, tandis que les MFCC sont une représentation compacte des caractéristiques spectrales.

Les chiffres qu'ils génèrent sont différents en raison de ces objectifs distincts et des étapes de traitement.

Image MFCC et Mel spectrogram

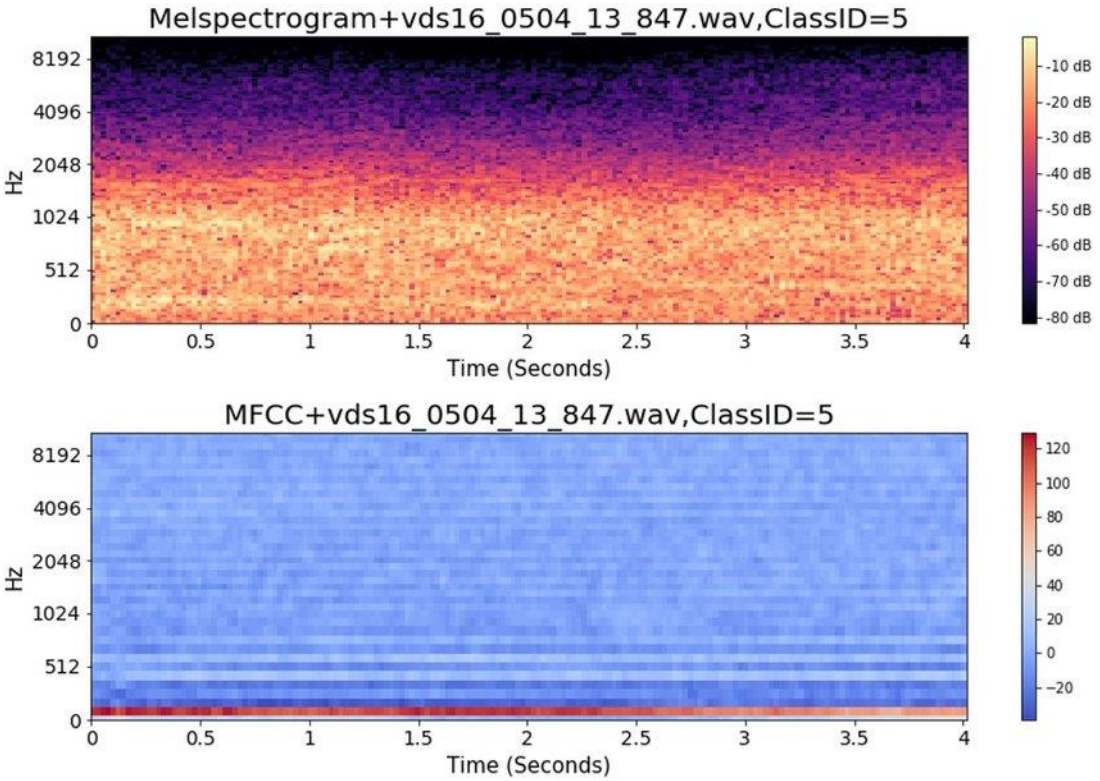
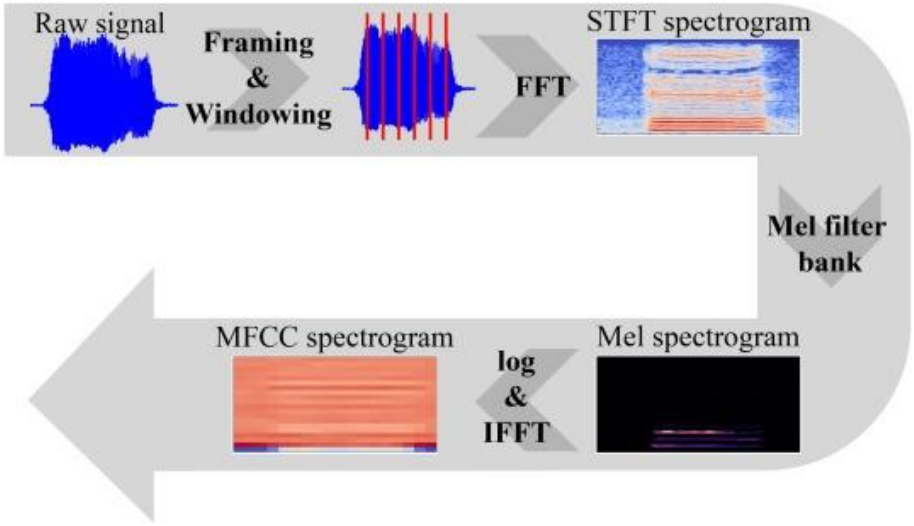


Image : represetation de la demarche utilisé pour passer d’un signal audio vers un MFCC :



Ensuite l'application de Neural Network sur ces données préparées.

Partie 2 : traitement de text

Pour la partie textuelle, il faut appliquer uniquement des traitements de NLP (Natural Language Processing) sur les phrases. Ces traitements comprennent principalement la tokenization et la lemmatisation :

Tokenization : c'est le fait de diviser une phrase en unités discrètes appelées tokens. Ces tokens peuvent être des mots individuels, des ponctuations, ou d'autres éléments significatifs tels que des chiffres ou des symboles. La tokenization permet de structurer le texte en éléments plus petits et facilite ainsi son analyse ultérieure.

Lemmatisation : c'est le fait e réduire des mots à leur forme canonique ou de base, appelée lemme. Cela implique de transformer les mots en leur forme principale, afin de normaliser le texte et de réduire la dimensionnalité du vocabulaire. Par exemple, les mots "manger", "mangeant" et "mangé" seraient tous réduits à leur lemme "manger".

Apres ça ,on est besoins d'appliquer un model de ML ,le model que j'ai choisie dans mon cas et le Naives bayer clasifier,…

Partie 3 : résumé automatique

dans cette partie il n'a pas de données a préparés ,le model et l'implementation de la manière de traitement et le seul besoin pour cette partie ,le model choisie pour cette detection est TF-IDF,c'est un model basé surla calcule de la fréquence des mot qui compose le documment ,afin de prendre uniquement les important phrase et les combiné pour obtenir le résumé.

Partie 4 : l'interface graphic

il exist deux étapes à faire ,la première et de créer un interface utilisateur et implementer le coté client avec ajax.

La deuxième étape et de créé un serveur node.js dans lequel on définit les route et utilisé child_process pour créer des sous routes où on va faire l'execution des notres models réalisé avec python. (après l'enregistrement des

moel ,en créé un code pythone dans le quelle en faire le loading de notre model et l'executé).

3.2 Mis à jour

Après une longue période de travail, j'ai rencontré des problèmes liés au traitement des fichiers audio. La génération des spectrogrammes MFCC à partir des fichiers audio .WAV a pris 12 heures d'exécution et nécessité 14 Go d'espace de stockage pour les spectrogrammes. De plus, la création du CNN a nécessité l'utilisation de TensorFlow, mais un module appelé distutils manquait. Après des recherches, j'ai découvert que ce module avait été supprimé de la dernière version de Python, la 3.12. J'ai donc essayé de passer à la version 3.11 où le module était encore disponible, mais j'ai rencontré une erreur lors de l'exécution du programme : "ImportError: DLL load failed: Une routine d'initialisation d'une bibliothèque de liens dynamiques (DLL) a échoué." J'ai essayé diverses méthodes pour résoudre ce problème, mais en vain. Finalement, j'ai décidé de poursuivre mon travail sans utiliser les fichiers audio, en me concentrant uniquement sur la détection de la langue et la résumé automatique de texte.

4. Planning et organisation du travail

4.1. Planning prévisionnel

Voici un tableau qui représente les taches, leurs durées et les dépendance entre ces taches :

Etudes sur le NLP et les technologie basic de machine learning T1	1 week	
etudes de base sur le language de programation python T2	1 week	
etudes de la bibliotheque NLTK et panas T3	4 days	T2
etude dela bibliotheque sktlearn T4	1 week	T2
Choisir l'algorithm de ML à appliquer pour la detection de langue. T5	1 week	T1
recherche de la base de donnes convenable pour la detection de langue T6	1 day	T5
Création du model de detection de langue T7	3 days	T2-T3-T4-T5-T6
Choisir l'algorithms de ML pour le résumé automatique T8	1 week	T8-4-3-2
Création de model de résumé automatique T9	3 days	
Création de l'interface graphic avec html et css(bootstrap) T10	2 hours	T10
Réalisation de coté client avec ajax T11	5 hours	
Réalisation des routes au cours de l'implementation de serveur node.js T1	2 hours	T7,T9
Création des scripts python pour exécuté les model enregistré T13	1 hour	T12,T7
La liaison entre le serveur et le model de detection de langue T14	2 days	T12,T9
La liaison entre le serveur et le model de résumé automatique T15	3 days	T7,T9
Teste des models T16	1 hour	T12,T13,T14,T15
Test de la connection entre le serveur et les models T17	1 day	T7,T9,T10,T11,T12,T13,T14,T15

Échéances :

La date au plus tard de chaque tache représente la date de départ de la dernière tâche qui dépend de cette dernière, plus d'information dans le diagramme de GANTT dans la partie conception

Suivi et mise à jour :

Plus d'information dans le diagramme de GANTT dans la partie conception

5. livrables attendus

Le résultat attendue, contenir un ensembles des composant regroupé pour fournir par la suite une interface qui sert a détecté la langue d'un texte fournie et faire un le résumé automatique.

Concernant le model de détection de langue, il est implémenté avec l'algorithme Naïve Bayer Clasifier .

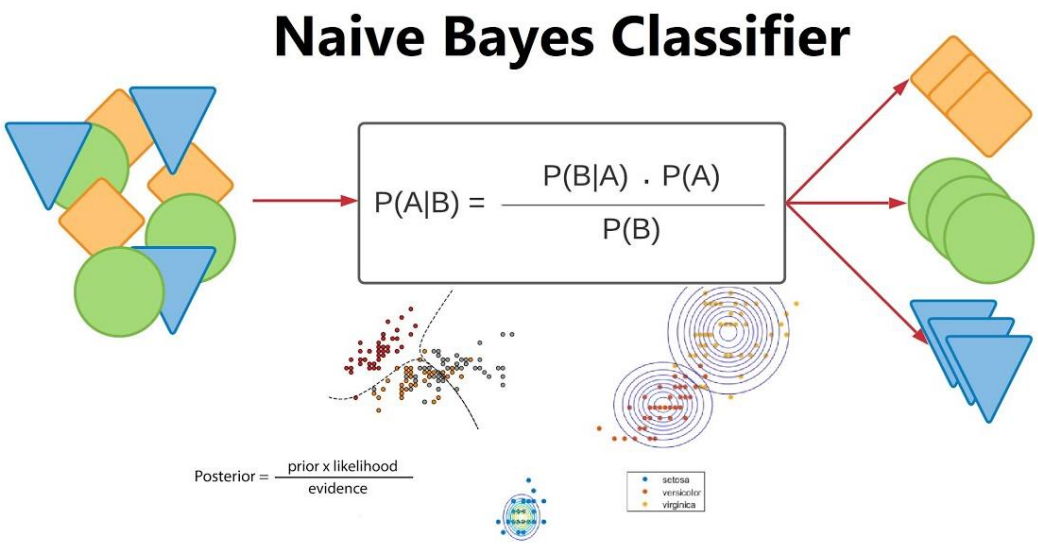
C'est quoi le model Naive Bayer Classifier :

D'après wikipedia :

> ``La classification naïve bayésienne est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires.``

Naive Bayes est un algorithme de classification simple et rapide, parfaitement adapté à des ensembles de données volumineux. Il est largement utilisé dans divers domaines tels que le filtrage de spam, la classification de texte, l'analyse de sentiment et les systèmes de recommandation. Son fonctionnement repose sur le théorème de Bayes, qui permet de calculer la probabilité d'appartenance à une classe inconnue à partir des probabilités conditionnelles des caractéristiques observées. Sa simplicité et son efficacité en font un choix populaire pour de nombreuses applications de traitement de données.

Voici une image explicativ de Naive Bayes Cassifier



Voici un exemple explicatif :

L'algorithme Naive Bayes est utilisé pour prédire si un e-mail est du spam ou non-spam. Il utilise une méthode simple basée sur la probabilité pour cette prédiction.

Imaginez qu’on a une boîte de réception remplie d'e-mails. Certains de ces e-mails sont des spams, tandis que d'autres sont des e-mails légitimes.

Pour entraîner le modèle Naive Bayes, nous examinons un grand nombre d'e-mails déjà étiquetés comme spam ou non-spam. Nous regardons les mots qui apparaissent dans ces e-mails et calculons la probabilité qu'un e-mail soit du spam en fonction de ces mots.

Par exemple, si nous remarquons que les mots "gratuit", "offre", et "promo" apparaissent souvent dans les e-mails étiquetés comme spam, nous attribuons une probabilité plus élevée à un e-mail contenant ces mots d'être du spam.Ensuite, lorsque nous recevons un nouvel e-mail, le modèle Naive Bayes examine les mots qu'il contient et calcule la probabilité que cet e-mail soit du spam ou non. Il prend en compte toutes les probabilités des mots et combine ces informations pour obtenir une probabilité globale.

Si la probabilité que l'e-mail soit du spam dépasse un certain seuil, alors le modèle le classe comme spam. Sinon, il le classe comme non-spam.

L’avantage de ce modèle c’est qu’il s’imple a implementé,et a comprendre ,aussi il ne naicessite pas une large quantité de donné pour un résultat exact et cela car en raison de son approche probabiliste simple et efficace, il suppose que les caractéristiques sont indépendantes les unes des autres, ce qui signifie qu'il ne nécessite pas autant de données pour estimer les probabilités conditionnelles des différentes caractéristiques.

J’ai essayé d’utiliser le model logisticregrition,mais même si l’accurcy est elevé 0.95 , il me donnée un résultat fausse ,voicie une capture de démenstration :

```
from sklearn.linear_model import LogisticRegression
ada = LogisticRegression()
y_pred = ada.fit(X_train,y_train).predict(X_test)
```
✓ 28m 34.8s

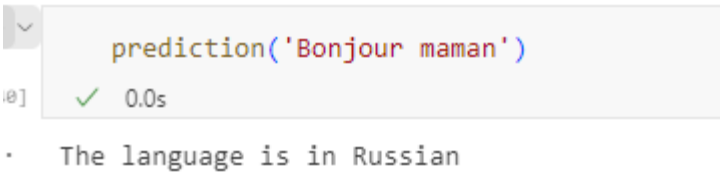
```
from sklearn.metrics import f1_score, confusion_matrix
ac = f1_score(y_test , y_pred, average='macro')
```
✓ 0.2s

```
print(f'Accuracy = {ac:.2f}')
```
✓ 0.0s

```
Accuracy = 0.95
```

```
def prediction(text):
    x = cv.transform([text]).toarray()
    lang = ada.predict(x)
    lang = le.inverse_transform(lang)
    print('The language is in',lang[0])
```
✓ 0.0s

```
prediction('Bonjour maman')
```
✓ 0.0s



La deuxieme étap et pour le model de résumé automatique ,le model qui j'ai choisie est le Tf-IDF

C'est quoi TF-IDF?

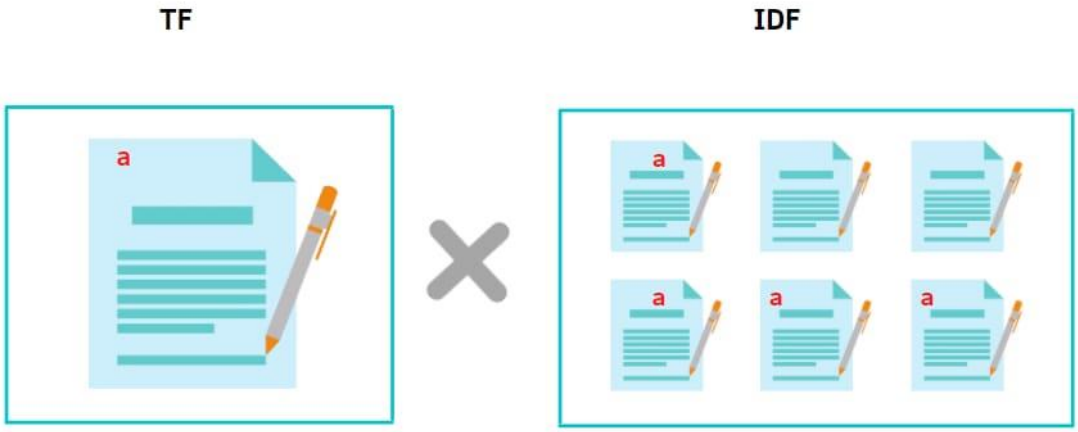
Abréviation de (Term Frequecy-Inverse Document Frequency C'est un model de ,Machine learning basé sur un technique de NLP ,utilisé pour evaluer l'importance des different mots dans une phrases.

le modèle TF-IDF est utilisé pour évaluer l'importance des mots ou des phrases dans un document afin de sélectionner les informations les plus pertinentes à inclure dans le résumé final . il exist plusieur mesur pour determiner la frequence de mot (TF),soit le calcule de nombre d'apparence de mot ,ou faire un dévision par rapport au numero des mot dans le documment,

par contre IDF représente l'inverse de la fréquence des documents (Inverse Document Frequency). Cette mesure évalue l'importance d'un terme dans l'ensemble du corpus de documents. Les termes qui apparaissent dans de nombreux documents ont une faible valeur IDF, mais les termes qui apparaissent dans peu de documents ont une valeur IDF élevée,et sont considéré comme les plus informatifs.

A l'aide de ça ,il évalue l'importance des mots ou des phrases dans le document et sélectionner les informations les plus pertinentes à inclure dans le résumé final en multipliant le TF et le IDF. Les termes avec les poids TF-IDF les plus élevés sont généralement sélectionnés pour représenter les points clés du document d'origine dans le résumé

J'ai le choisie puisque c'est un model car il est le plus adapté par la résumé automatique grace au plusieurs résons,la premiére c'est qu'il est adapté pour variété des domaines des documents ,aussi il ne s'ainteress pas uniquement par les mot les plus auccurent dans le text mais aussi pour les râres, de plus il est facile a implémenté , ne nécessite pas de modélisation complexe ou d'apprentissage supervisé, ce qui peut rendre le processus plus rapide et plus efficace.



Le TF représente la fréquence de mot par rapport au document de text ,et le ODF représente la fréquence de mot par rapport à chaque partie de document (text)

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

L'aspes mathématique de (TF-IDF).

Maintenant ,l'interface utilisateur :

Pour l'interface graphic,il doit contenir uniquement une deux zone de text l'une avec ecriture désactivé,puis de boutton,l'une pour l'action de detection de langue et l'autre pour la summarisation.

Sur les boutton,il y a un listner sur lequelle le code ajax est créé pour envoyé les données naicessaire au serveur et retourné le résultat d'après ce dernier.

Puis ,pour le coté serveur,les route sont définie,notament deux routes demandé , les deux pour la methode post ,la première pour la boutton get langugae,il doit retourn les donné envoyé après la click sur le boutton (code implémenté avec ajax),puis interagie avec le model enregistré de detection de la langue,et envoyé le résultat de model ain de l'afficher pour l'utilisateur.

La deuxieme et pour le boutton get summary ,dans cette erniere il faut prend en considération si le boutton e getLanguage et clicker ou non,car la fonction qui represent le modele naicessite deuxx parametre le premier et la langue du tet inseré et le deuieme est le text,si le boutton déjà cliqué alors la langue obtenue de la zone ou la langue est afficher ,sinon,il faut faire l'appel au model de detection de langue ,puis obtene la langue et l'utilisé comme parametre avec le texte pour obtenir le résumé automatique.

Et enfin ,le test final pour validé le résultat final.

Conclusion

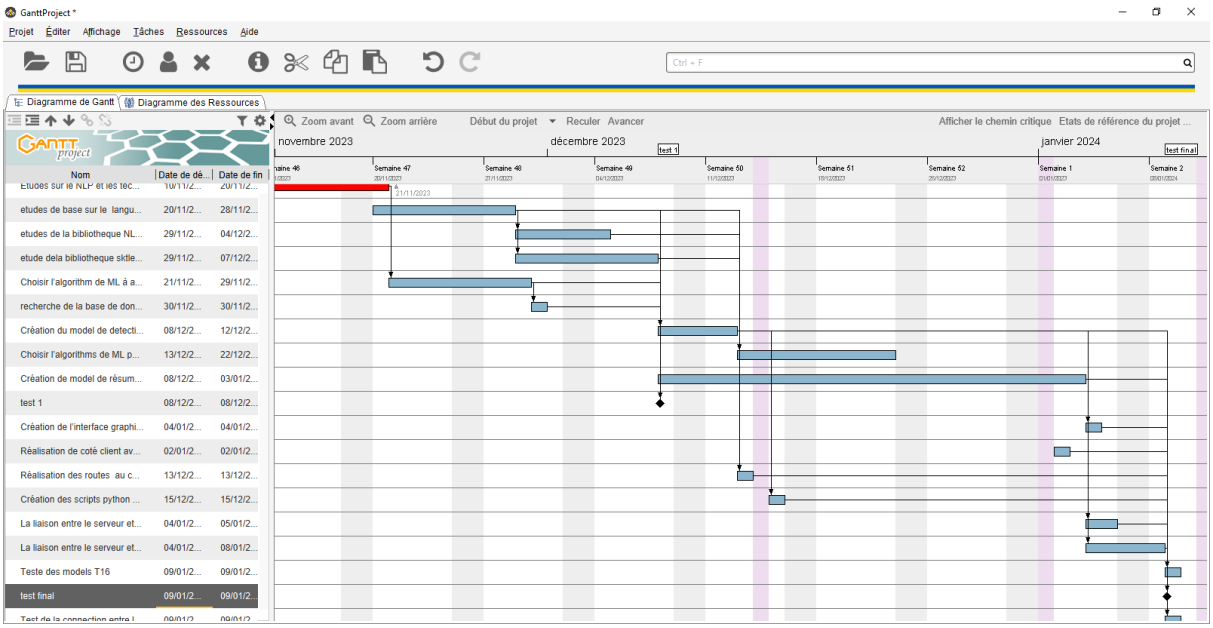
je souligne ,comme conclusion que le projet vise à répondre à un besoin essentiel dans le domaine du traitement automatique du langage naturel en fournissant une solution efficace pour la détection de la langue et le résumé automatique de texte. En utilisant des technologies telles que le Naive Bayes classifier et le modèle TF-IDF

j'ai essayé de bien planifier le projet ,et determiner les taches principal afin de ne pas raté la durée de chacune des taches depuis l'exploration des technologies de base jusqu'à la mise en œuvre des modèles et la création de l'interface utilisateur.

Je reste déterminé à poursuivre mes efforts et à livrer un rapport complet et satisfaisant, répondant aux attentes de toutes les parties prenantes impliquées.

II. Conception du projet

Image représente le diagramme de gantt de projet :



Imag represente le diagramme de pert de projet :

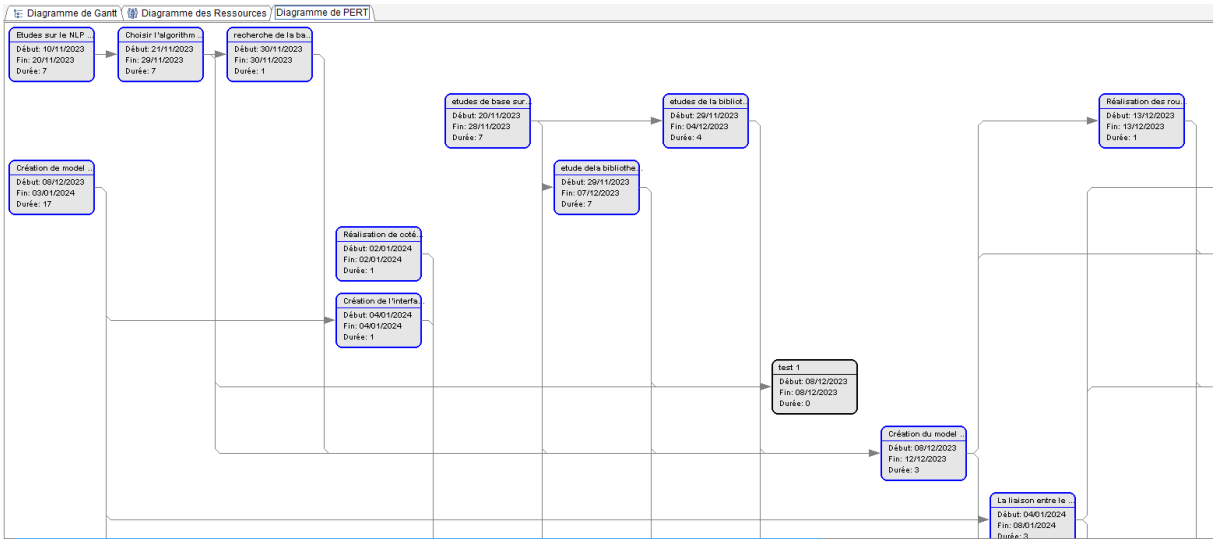
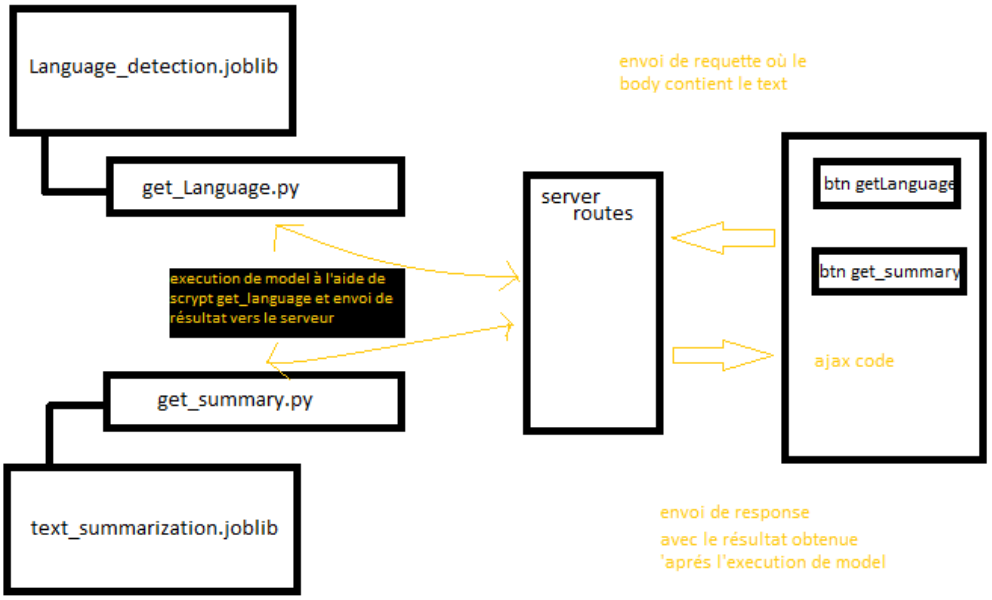


Image représente une simulation des different interaction des composant de plateforme :



III. Outils est technologies utilisé

La réalisation de ce projet naicessite une large gamme des outils et technologie :

Environnement de développement intégré (IDE) :

J’ai pas essayé d’utilisé un diffeent IDE que visual studio code puisque c’est le plus populaire,la seul chose que j’ai ajouté sont les extention,la première et de python, suivie par l’extention e jupyter afin ‘obtenir un envirenement similaire de jupyter notebook pour les ficher avec l’extention .ipynb

Un Jupyter Notebook est un environnement de développement interactif qui permet d'écrire et d'exécuter du code en Python (ou dans d'autres langages comme R ou Julia) de manière progressive, combinant des cellules de code, de texte, d'images et de visualisations dans un seul document.

Gestionnaire de packages :

Pour les gestionnaire des package,j’ai utilisé pip puisqu’il est le gestionnaire de packages standard pour Python.

Bibliothèques Python :

Il exist nombreux bibliothéque utlisé an mon projet ,j’ai déjà les mentionné dans la partie de 1.4 de Chapitre I ‘Etat actuel des resources et fonctionnement du préexistant,mais je les rementioné avec une bréf explication :

NumPy : utilisé pour le calcul numérique.

pandas : utilisé pour la manipulation et l'analyse des données.

Matplotlib: utilisé pour la visualisation des données.

scikit-learn : utilisé pour l'apprentissage automatique.

NLTK (Natural Language Toolkit) ou spaCy : utilisé pour le traitement du langage naturel (NLP).

Environnement virtuel :

L’envirenement virtuel et l’outil principal qui permet d'installer les packages Python spécifiques pour un projet sans interférer avec les autres projets ou le système global. Dans mon cas j’ai utilisé conda (Anaconda),conda est un outils polyvalent,il est pas utilisé uniquement pour l’envirenement virtuel mais aussi pour la gestion des packages et plus,j’ai choisie de l’utilisé car il gère

automatiquement les dépendances entre les packages, ce qui garantit que toutes les dépendances requises sont correctement installées et compatibles,aussi c'est facile a utilisé et simplifie la création, la gestion et l'utilisation des environnements virtuels avec quelques commandes simples.

Les algorithms de ML et les techniques de NLP :

NaiveBayerClassifier : Le Naive Bayes Classifier est un algorithme de classification supervisée simple mais puissant, souvent utilisé pour la classification de texte, telle que la détection de spam, l'analyse de sentiments, ou la catégorisation de documents. Il est basé sur le théorème de Bayes et suppose que les caractéristiques sont indépendantes les unes des autres.

TF-IDF : TF-IDF est une méthode utilisée pour évaluer l'importance d'un terme dans un document par rapport à un corpus de documents. Elle prend en compte à la fois la fréquence d'apparition d'un terme dans un document (TF) et l'inverse de sa fréquence d'apparition dans l'ensemble des documents du corpus (IDF).

Vous pouvez voir plus d'information dans la partie 4 chapitre 1 (livrables attendue).

HTML : c'est le language principal pour le coté static des sites web.

Bootstrap : c'est un framework open source développé qui fournit une collection d'outils et de composants front-end prêts à l'emploi pour le développement rapide et la conception de sites web et d'applications web réactives,il permet de personnaliser facilement l'apparence de site avec les thèmes prédéfinis.

Ajax (Asynchronous JavaScript and XML) : c'est une technique de développement web qui permet de créer des applications web interactives et dynamiques en utilisant JavaScript et des requêtes HTTP asynchrones,c-à-d mises à jour asynchrones et partielles des contenus web sans rechargement complet de la page.

Node.js : un environnement d'exécution JavaScript côté serveur, construit sur le moteur JavaScript V8 de Chrome, permet d'exécuter du code JavaScript côté serveur, et utiliser pour créer des applications serveur, des API, des applications web en temps réel .. .

child_process : un module intégré à Node.js qui permet d'exécuter des processus externes depuis une application Node.js. et fournit plusieurs

méthodes pour créer, contrôler et communiquer avec ces processus externes,dans notre cas on a s'interessé par ma methode spawn : qui est utilisée pour démarrer un processus externe et obtenir un flux de sortie (stdout) et un flux d'erreur (stderr) pour l'execusion de notre script de l'execution des model python créé.

Express : Express est un framework web minimaliste et flexible pour Node.js,qui simplifier le processus de création d'applications web et d'API. Il fournit un ensemble de fonctionnalités robustes pour la gestion des routes, la gestion des requêtes et des réponses http.

Body-parser : un middleware pour Express qui permet de récupérer les données envoyées dans le corps des requêtes HTTP, telles que les données de formulaire ou les données JSON. Il analyse le corps des requêtes et le transforme en un objet JavaScript utilisable par l'application.

IV. Réalisation

l'envirenement de travail :

Premierement ,j'ai commencer par l'installation de python,j'ai travailler avec la version 3.11 puisqu'il contient des modules pas encore supprimé par rapport au v3.12,pui j'ai télechargé anaconda afin d'avoir un envirenement virtuel de python,puis j'ai ajouté pip et python dans Path avec les variables d'envirenement car j'ai rencontré u problém de commande introuvable pour pip et python,puis j'ai telecharger l'extention python et jupyter pour obtenir un envirenement similaire que jupyter notebook.

model de detection de langue :

Au début,j'ai commencer par l'ajoue de fichier csv qui contient lles données que je serai utilisé pour la partie de training et pour la partie de testing,puis j'ai commencer a importé les package naicessaires pour la réalisation de ce model,qui sont pandas,numpy,nltk,re

Puis j'ai chargé les données a partir du fichier csv et j'ai passé vers la préparation de cette derniére,la premier etape et de supprimer les caractére spéciaux et les chiffre,et en mettre tout le text en miniscule ,aprés j'ai encodé

les langue en valeurs numériques,et j'ai utilisé ces valeurs comme étiquette , chaque ligne dans le tableau qui contien les donné sera associée à la valeur numérique de la langue de sa colonne correspondante comme étiquette,et comme ça ,j'ai créé plusieurs class,chacune pour un language spécifique.

L'etape suivant et de convertir les phrases en vecteurs,et cela basé sur le CountVectorize(),premierement CountVectorizer commence par construire un vocabulaire ''Bag-of-Words,, à partir de toutes les mots uniques dans l'ensemble des phrases. Chaque mot unique est considéré comme une "caractéristique" du texte,puis il attribue un index à chaque mot du vocabulaire ,et comme cela,chaque phrase est convertie en vecteur dont les nombres sont les index attribuer au mots qui composent cette phrase,la même procédure est appliqué a toutes les phrases de la base de données.

Après cette étape ,les données sont préparé ,je les divisé en donné de training et données de testing,pour la partie training 80% des données sont fournie ,par contre pour la partie testing ,uniquement 20% sont fournie,j'ai utilisé cette demarche car Plus l'ensemble d'entraînement est grand, plus le modèle aura de données sur lesquelles apprendre, ce qui peut donne come résultat une meilleure performance du modèle .

Après j'ai initialisé la clasification Multinominal de Naive BayerClassieir ,et j'ai trainer ce model avec les vecteurs des phrases de training et les index des langue avec la methode fit, je pas vers l'evaluation sur l'ensemble de test

Sur le 20% des données que j'ai laissé pour la partie de testing ,et pour assure que chaque classe a le même poids dans le calcul du score F1 global,jai utilisé verage='macro',et comme cela les classes plus petites serons avoir le même poids que les classes plus grandes dans le calcul du score global (il sera moyenné pour obtenir un score global) , et j'affiche à la fin le score ,puis j'enregistre le model sous à l'aide de dump.

Test de model de detection de langue :

Concernant la partie testing,il est inclue dans la réalisation de projet (l'étape avant dérnière),mais après j'ai essayé de testé avec des different phrase ou les mot sans les même pour les deux langue,ce que j'ai remarqué que si je onne un mot qui est le même pour deux langue (anglais et autre langue) sans article,ce dernier sera considéré comme englais ,le cas que j'ai donnén'a pas un réponse exact mais aussi la réponse donné par le model rest correct puisqu'il n'a pas d'article pour spécifier.

model de résumé automatique

Pour ce model,j'ai choisie d'utilisé la methode TF-IDF,j'ai déjà parlé de cette dérnière dans la partie 4 chapitre 1 (livrable attendus).

Premièrement je faire le stemming pour réduire les mot a leur format de base et éliminé les stop words ,aussi igore les mot avec un nombre de charactére inférieur à 3 puisq'il ne serons pas contient des info de base que je cherche dans le documment ou le text ,puis je créé une table de fréquence de ce mot en fonction des phrases qui compose le text,j' utilise la methode de NLTK sent_tokenize pour capté les phrases de text et je réalise une matrice de fréquence des mot pour chaque phrase,ensuite je pass vers le calcule de TF,avec la matrice des mot pour chaque phrase que j'ai fait je calcule la matrice TF (Term Frequency) en divisant chaque fréquence par le nombre total de mots dans la phrase,je garde ce calcule et je pass vers le calcule de nombre de phrases contenant chaque mot ,et cela a l'aide de la matrice de fréquece des mot pour chaque phrase que j'ai déjà réalisé,puis je pass vers le calcule de l'IDF,l'iDF ,et par la suite je calcule le score en fesons une somme de TF et IDF , et j'utilise ce résultat pour calculer le score des phrases ,ensuite je calcule le scors moyenn des phrases et je determine un seuil, le seuil represente le score moyen des phrases multiplier par un facteur spécifique ,dans mon cas j'ai utilisé 1.15 puisqu'il est recommendé,a la fin ,pour le résumé automatique, en choisissant uniquement les mot avec un score supérieur à ce seuil ,en combinant ces phrases entre eux et en obtenir par l suite un résumé de notre texte /documment .

test de model do résumé automatique

J'ai testé avec deux fichiers textuelles et j'ai obtenue un résultat satisfant ,dans lesquelle le nombres des mot et réduit ,et le text donne une idée génèral de l'encienne documment détallé.

création de l'interface graphic

Pour l'interface graphic,j'ai utilisé premierement htlml pour créer deux zone de text la premier j'ai gardé la notion de l'ecriture activé mais je la supprimé de la de l'autre, aussi deux boutton une pour le résumé et l'autre pour la langue ,et une label 'langue, suivie par une balize p vide ou je serai inséré la langue detécté.

Ensuite j'ai utilisé selement bootstrap pour donner une taille grande au zone de text et un style pour les boutton ,et alignement au centre.

implémentation de coté Serveur(Node.js)

Au coté serveur,j'ai réalisé deux routes ,une pour le résumé et l'autre pour la detection de la langue.

Premiérement j'ai importé tous ce qui est naicessaire,les moduls,les middlewear,et ainsi de suite,puis j'ai pasé vers la création des routes .

- Pour la route detection de langue('/getLanguage') :

C'est une route post avec le chemindéja mentioné,premierement j'ai Récupèré le texte envoyé dans le corps de la requête POST à l'aide de req.body.text, puis j'ai Crée un processus enfant dans lequelle j'execute le script Python language_detection.py,. ce script import le model que j'ai déjà enregistré dans un fichier .joblib et execute les methodes correspond a ce model (j'ai fais la même chose pour le résumé). L'eecution de ces scrypt est fait à l'aide de la méthode spawn du module child_process que j'ai importé précédament et j'envoie le texte récupéré dans le corps de la requête via stdin pourqu'il puisse utilisé et je termine l'écriture dans l'entrée standard du processus Python. Et je commence a ècouté les données envoyées par le processus Python sur sa sortie standard qui représent dans ce cas la langue de texte fournie,

Et en fin Envoie une réponse JSON au client(implémenté par ajax) contenant la langue détectée.

- Pour la route de résumé ('getSummary') :
- Aussi c'est une route post avec le chemin que j'ai mentionné, premièrement je faire le feth ala route getLanguage afin d'ontenir la langue de ce text fournie puis je crée un processus enfant pour exécuter le model de summarisation et j'envoi le texte récupéré de body de requête plus la langue detecté après l'appel de la route getLanguage,puis j'obtenir mon résultat de l'exécution de model et j'envoi la réponse au client qui est dans ce cas un summary de text.

test de coté serveur

J'ai utilisé une extension dans vs code qui me permet de généré des requêtes ,j'ai défini le port de l'écoute et l'adress localhost et j'ai essayé avec un text,et heuresement ,j'ai obtenue un response qui contient dans le boy la langue detecté.

J’ai fait la même chose pour la partie de résumé automatique.

implémentation e coté client (Ajax)

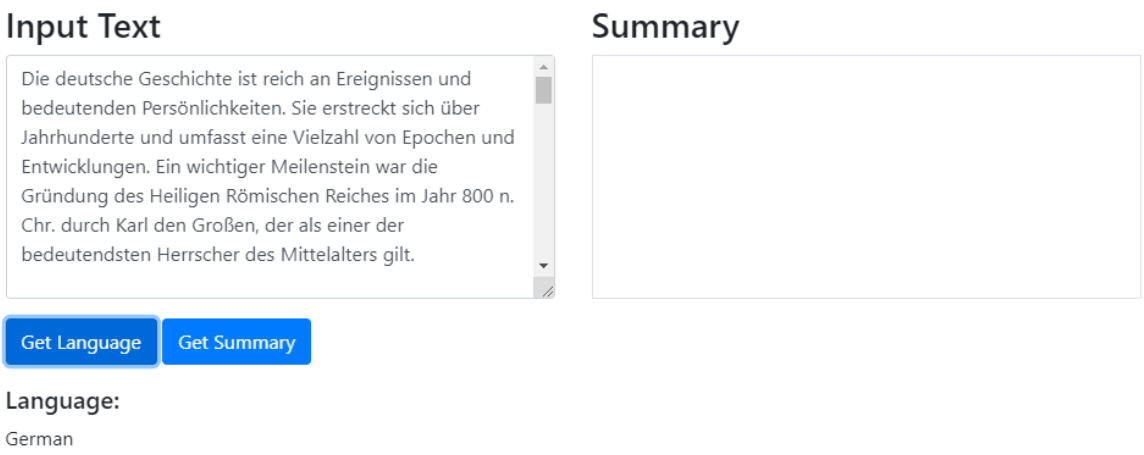
Pour cet partie ,j’ai déjà donné des id au deux boutons et au zones de texte ainsi que le div ou la langue détecté sera affiché ,je créé des variable pour obtenir ces composant d’après ces id ,puis j’utilise la methode addeventListener de l’évènement click ,et j’ai défini la fonction à exécuté ,pour les deux boutons (get language et get Summary,la seul chose que je serai fait et de préparé une requette httl vers le serveur node.js où j’ai déjà défine les deux routes,cette requette utilise la méthode post et envoi dans le body le texte fournie par l’utilisateur dans la zone de texte.

Voici une capture d’écran pour visualisé comment ça marche la détection et la génération de résumé :

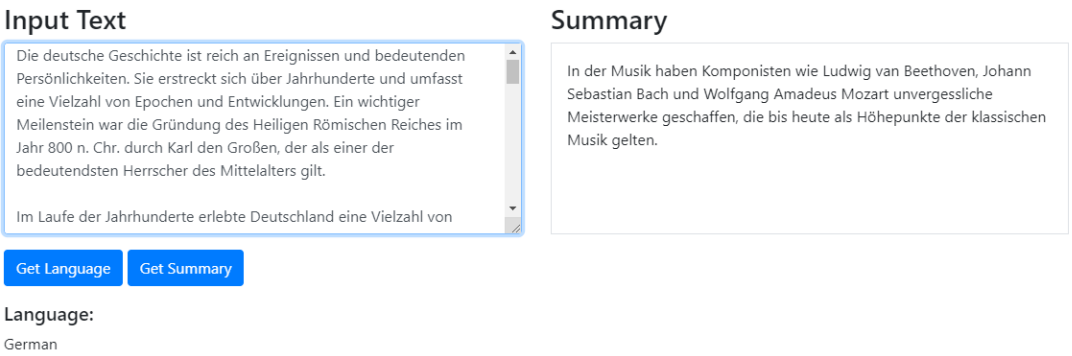
L’interface :

Test avec un texte en allemand :

Détection de langue :



Génération de résumé :



V. Conclusion

Dans le cadre de mon projet de fin d'études, j'ai développé une plateforme de détection de langue et de résumé automatique de texte dans le but de faciliter la compréhension et l'analyse de documents dans différentes langues. Mon objectif principal était de créer un outil polyvalent capable de traiter les textes provenant de sources diverses et de générer des résumés concis et informatifs.

Pour atteindre cet objectif, j'ai utilisé une approche basée sur le traitement automatique du langage naturel (NLP) et l'apprentissage automatique (ML). J'ai commencé par collecter des données textuelles dans différentes langues, puis j'ai procédé à un prétraitement rigoureux, comprenant le nettoyage des données, la tokenisation et la suppression des stopwords.

En ce qui concerne la détection de langue, j'ai développé un modèle basé sur des caractéristiques linguistiques telles que la fréquence des mots et la structure syntaxique. Pour le résumé automatique, j'ai adopté une approche basée sur la fréquence des termes-inverse du document (TF-IDF) pour identifier les phrases les plus importantes dans le texte. En utilisant cette méthode, j'ai pu générer des résumés pertinents en préservant l'essence du contenu original.

Bien que mon système ait montré des performances encourageantes, il présente encore certaines limitations. Par exemple, il peut rencontrer des difficultés avec les langues moins représentées dans mes données d'entraînement, et la qualité des résumés peut varier en fonction de la complexité du texte source. De plus, la détection de langue ne couvre pas toutes les langues du monde, mais seulement quelques-unes selon la base de données disponible.

De plus, je n'ai pas pu aller plus loin dans le traitement des audio en raison d'un problème lié au package TensorFlow qui m'a pris beaucoup de temps, mais malheureusement sans résultat.

Pour les futures directions de recherche, j'envisage d'explorer des techniques de ML pour améliorer la précision de la détection de langue, ainsi que d'intégrer la notion de traitement des audio en utilisant des méthodes de deep learning et de réseaux neuronaux. De plus, je souhaite améliorer la qualité des résumés automatiques et découvrir d'autres techniques à l'avenir.

Ce projet a représenté une étape importante dans ma carrière dans le domaine du traitement automatique du langage naturel. Je tiens à exprimer ma gratitude à mon encadrant pour son soutien et ses conseils tout au long de ce projet, ainsi qu'à l'ESTF pour m'avoir donné cette opportunité d'apprentissage et de développement.

Annexes :

https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne

https://www.oracle.com/ch-fr/artificial-intelligence/machine-learning/what-is-machine-learning/

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/#:~:text=Using%20scikit%2Dlearn-,What%20is%20TF%2DIDF%3F,%2C%20relative%20to%20a%20corpus).

https://www.ibm.com/topics/natural-language-processing

https://www.analyticsvidhya.com/blog/2021/04/role-of-machine-learning-in-natural-language-processing/

https://ieeexplore.ieee.org/document/9955539

https://fr.mathworks.com/matlabcentral/answers/597799-difference-between-fft-and-pspectrum

https://www.science.gov/topicpages/f/fft+spectrum+analyzer.html