

Rapport de Mini Projet



Assistant d'aide à la rédaction médicale basé sur la reconnaissance vocale et l'IA.

Réalisé par :

Oueslati Beyrem, Gmiza Mohamed Aziz, Zakhama Yasser

Matière :

Mini-projet en intelligence artificielle

Présentation du projet

- **Problématique**

Dans le domaine médical, les médecins passent beaucoup de temps à rédiger des rapports, comptes rendus et notes cliniques après leurs consultations. Cette tâche administrative peut être longue, répétitive et source d'erreurs, réduisant le temps qu'ils peuvent consacrer aux patients.

Dans ce contexte, nous développons un assistant IA « Doc Voice » spécialisé dans la dictée médicale . Grâce à des modèles de reconnaissance vocale avancés (Whisper) il va transformer en temps réel la parole en texte et garantir la précision en intégrant la détection des termes médicaux en se basant sur une base de données de termes médicaux. Tout cela facilite la structuration des rapports et permet aux médecins de gagner un temps précieux, réduire les erreurs et se concentrer davantage sur la prise en charge des patients.

Lien GitHub

<https://github.com/beyremweslati/docvoice-speech-to-text>

Fonctionnalités

Besoins Fonctionnels

- **Enregistrement vocal** : Le système doit permettre à l'utilisateur (médecin) d'enregistrer sa voix.
- **Transcription automatique** : Le système doit convertir la voix en texte en temps réel ou après l'enregistrement.

- **Intégration du vocabulaire médical** : Le modèle doit utiliser une base de données de termes médicaux pour améliorer la précision de la transcription.
- **Génération automatique du rapport** : Le texte transcrit doit être structuré selon un modèle de rapport médical)
- **Édition et validation** : Le médecin doit pouvoir corriger et valider le rapport avant de le sauvegarder
- **Sauvegarde et exportation** : Le rapport final doit être sauvegardé dans la base de données et exportable au format Word (.docx) ou PDF.

Besoins Non Fonctionnels

- **Performance** : Le système doit offrir une transcription rapide et fluide, avec un temps de latence minimal.
- **Scalabilité** : Le système doit pouvoir gérer plusieurs utilisateurs simultanément sans perte de performance.
- **Portabilité** : L'application doit être accessible via navigateur web et facilement déployable sur serveur.
- **Maintenabilité** : Le code doit être bien structuré et documenté pour permettre les futures améliorations.

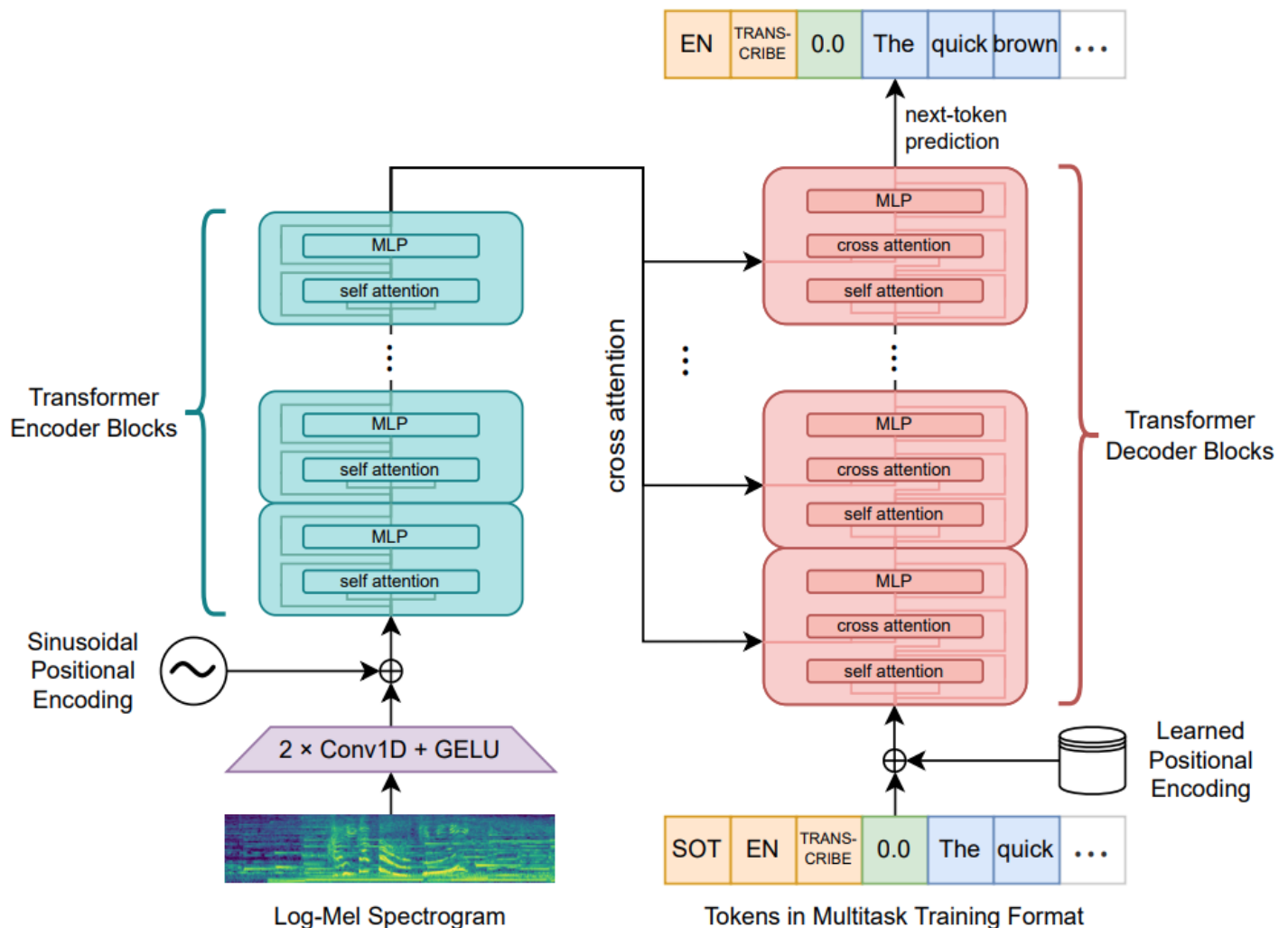
Technologies utilisées

Langage de programmation : Python.

Environnement du travail : Jupyter Notebook.

Technologie	Rôle
Elevenlabs	Service de synthèse vocal basé sur l'IA permettant de générer un dataset audio à partir de texte
Whisper	Modèle de reconnaissance vocale d'OpenAI utilisé pour traduire les fichiers audios.
Pandas	Lecture et traitement des fichiers .CSV contenant le dataset des termes médicaux.
OS	Travailler avec le système d'exploitation : fichier, dossiers, chemins.
WAV	Lire/écrire des fichiers audio générés par elevenLabs au format WAV
Scipy.io.wav File	Lire les fichiers .wav pour les convertir en une forme utilisable par Whisper
Json	Module python pour lire et écrire des fichiers json contenant des datasets (audio +text)
PyTorch	Framework deep Learning utilisé pour faire tourner le modèle whisper sur CPU ou GPU.
Transformers	Charger le modèle Whisper pré entraîné, préparer les données audios et transformer l'audio en texte.
Scikit-Learn	Algorithme du machine Learning permettant de séparer les données en train /validation / test.
Jiwer	Calculer les métriques d'erreurs en speech-to-text (WER, CER)
LoRA	Méthode de fine-tuning légère et plus rapide qu'un fine-tuning complet, elle permet d'adapter le modèle Whisper au domaine médical en entraînant une petite partie.
Regex(re)	Identifier les couches linéaires nécessaire du LoRA
Streamlit	Framework Python permettant de créer facilement des interfaces web interactives. Utilisé pour rendre le modèle IA (Whisper) accessible via une interface web
FPDF	bibliothèque Python de génération de fichiers PDF.

Architecture du modèle Whisper :



Le modèle Whisper est une architecture de type **Encoder-Decoder (Encodeur-Décodeur)** basée sur des blocs de **Transformer**, entraînée pour effectuer des tâches multiples comme la transcription et la traduction.

1. L'encodeur : (partie d'analyse d'audio)

Quand le modèle Whisper commence à écouter le son du docteur, l'encodeur traite le signal audio pour en extraire des caractéristiques

- **Entrée audio** : Le signal audio est d'abord transformé en Spectrogramme de Mel, c'est une représentation des fréquences au fil du temps
- **Couches Initiales** : Le spectrogramme passe par deux couches de Convolution 1D (2*Conv1D) avec une fonction d'activation GELU. Ces couches extraient des caractéristiques de base (comme les changements soudains de la fréquence, les montées et descentes de volume...).
- **Encodage Positionnel Sinusoïdal** : Pour préserver l'information temporelle (l'ordre de l'audio), des Sinusoïdal Positional Encoding sont ajoutés à la sortie des couches de convolution.
- **Blocs de Transformer** : La séquence de caractéristiques enrichies est traitée par une pile de Transformer Encoder Blocks chacun contenant :
 - **Self-Attention** : permet à l'encodeur de regarder tous les fragments de son à la fois et de déterminer quelles parties de l'audio sont les plus importantes les unes par rapport aux autres pour décoder un mot. (en attendant le son d'un mot au début , l'attention lui donne une grande importance pour comprendre le mot à la fin).
 - **MLP (Multi-Layer Perceptron)** : Applique des transformations non-linéaires.
- **Sortie** : L'encodeur produit une séquence de **vecteurs contextuels** (résumé numérique qui contient des informations sur la nature du son, contexte et sa position dans la séquence) qui seront transmis au décodeur via cross attention.

2. Le décodeur (partie qui génère le texte)

Le décodeur utilise les caractéristiques de l'audio pour générer la séquence de *tokens* de texte.

- **Entrée de Texte (Tokens) :** Le décodeur reçoit une séquence de tokens de texte. Cette séquence commence par des *tokens* spéciaux, par exemple :
 - **SOT (Start of Transcript)** : Indique le début de la séquence.
 - **EN** : Spécifie la langue cible (l'anglais).
 - **TRANSCRIBE** : Spécifie la tâche à effectuer (transcription)
 - **0.0** : il indique au modèle qu'il doit commencer à écrire pour le son qui démarre à 0 seconde, pratique pour segmenter les longs enregistrements.
 - Viennent ensuite les tokens de texte générés précédemment (ex : "The quick").
- **Encodage Positionnel Appris :** Pour les tokens de texte, des Learned Positional Encoding sont ajoutés pour indiquer leur position.
- **Blocs de Transformer :** La séquence d'entrée de tokens est traitée par une pile de Transformer Decoder Blocks, chacun contenant trois mécanismes d'attention :
 - **Self-Attention (Masquée)** : Similaire à l'encodeur, mais masquée pour que la prédiction d'un token ne se fasse qu'en fonction des tokens précédents.
 - **Cross Attention** : C'est ce qui permet au Décodeur, à chaque nouveau mot qu'il écrit, de pointer vers la partie exacte du "résumé" audio qui correspond à ce mot.
 - **MLP (Multi-Layer Perceptron).**
- **Sortie :** Le décodeur effectue la prédiction du prochain token complétant ainsi la séquence de sortie.

Les étapes d'exécution :

Notre projet est constitué de cinq fichiers python :



1. generateAudio.py :

Ce code lit un fichier CSV contenant le dataset des termes médicaux, et génère pour chacun un fichier audio (format PCM) et un fichier texte correspondant à l'aide de l'API ElevenLabs.

2. [wavConverter.py](#) :

Ce code parcourt les fichiers audios au format PCM puis les convertit en fichiers WAV.

3. [initialAnalysis.ipynb](#) :

Ce code crée un dataset en liant chaque fichier audio à sa transcription réelle puis l'enregistre au format JSON. Ensuite utilise Whisper pour générer les transcriptions, puis évalue la précision des prédictions avec les métriques WER et CER.

4. [fineTuning.ipynb](#):

Ce code entraîne le modèle whisper en appliquant LoRA afin d'améliorer la transcription audio-text sur le domaine médical.

5. [App.py](#)

Ce code crée une interface web avec Streamlit dans laquelle on peut enregistrer le vocal et le transcrit automatiquement en texte grâce à Whisper puis génère le rapport médical.

Exemple d'exécution :

