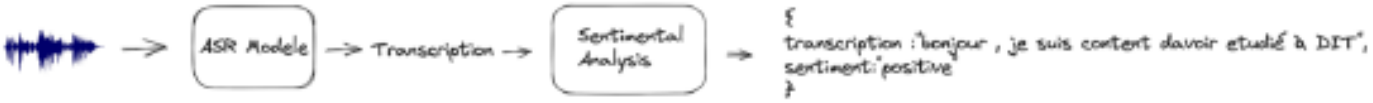
KUDAWOO Séna Kodjovi – Master 2 IA

Rapsport détaillé de projet de Deep Learning

Apercu du projet



Inference Architecture

Question 1 : Mise en place d'un modèle ASR (Automatic Speech recognition)

Objectif : Utiliser un Modèle Français sur le Hub de Huggingface en inférence pour convertir des enregistrements audios en texte.

Modèle choisi	*Whisper-large-v3*
Limitations	
Longueur Maximale des Séquences	Les modèles Whisper traitent les données audio en segments appelés mél-spectrogrammes. La longueur maximale de ces spectrogrammes est une contrainte importante. Pour Whisper, la longueur maximale des séquences est généralement limitée à 30 secondes d'audio.
Dimensions des Spectrogrammes	Les spectrogrammes générés à partir de l'audio doivent avoir des dimensions spécifiques. Whisper utilise des spectrogrammes de taille fixe, et le modèle attend une longueur de séquence fixe dans l'entrée.
Mémoire et Performance	La mémoire GPU ou CPU disponible peut limiter la taille effective des fichiers audio qu'on peut traiter en une seule fois. Les fichiers audios plus longs peuvent nécessiter plus de mémoire pour être traités en entier.
	Solutions implémentées
Découpage de l'Audio:	Nous avons défini une fonction qui divise les fichiers audio longs en segments plus courts (de 30 secondes) avant de les passer au modèle. Cela nous permettra de rester dans les limites de taille et de traiter chaque segment indépendamment.
Dataset de banc d'essai	- ***Audio :*** *Tsh/data/train/* - ***Transcription :*** *Tsh/train.csv*
Code	*speechTranslate.py*

Question 2 : Mise en place d'un modèle ASR (Automatic Speech recognition)

Objectif : Utiliser le modèle ASR précédent pour effectuer une transcription automatique d'un enregistrement audio fourni. Ensuite, effectuer une Analyse de Sentiment sur la transcription générée.

Pour atteindre cet objectif, nous allons suivre les étapes suivantes :

0. **Prérequis :** Modèle ASR (Whisper-large-v3)

1. **Etape 1 :** Pour l'analyse de sentiment, nous allons personnaliser le modèle de traitement du langage naturel (NLP) - BERT à partir de la dataset https://www.kaggle.com/datasets/djilax/allocine-french-movie-reviews qui classifie la polarité des sentiments d'un large nombre de commentaires à 0 ou 1 selon que cette dernière est négative ou positive.

Le modèle customisé sera entrainé via le notebook de Kaggle en raison des ressources en GPU nécessaires. Le résultat des poids du modèle sera sauvegardé dans notre dossier de travail

CODE	***SENTILYSIS.PY***
HYPERPARAMETRES	*N_classes = 2* *Bacth_size = 8* *Epoch = 1*
RESULTAT	`100%||| 2500/2500 [02:11<00:00, 19.08it/s]` `Train loss: 0.042462299009860725 | Valid loss: 0.2559778641391546 | Valid Accuracy: 0.8883999586105347 | Test loss: 3.052528227162361 | Test Accuracy: 0.11109999567270279`
MODELE	`my_custom_bert3.pth`

2. **Etape 2 :** A ce stade, nous disposons des deux modèles prêts à être déployés de sorte qu'à partir d'un fichier audio, on puise retourner une transcription en français, puis l'analyse du sentiment (***positif*** ou ***négatif***) contenu dans le dit fichier audio.

Pour le déploiement, la librairie FastAPI. Afin de tester le fonctionnement de l'API, nous utilisons l'application **Postman** via des requêtes `POST`.

Attention, pour le test vous devez télécharger et installer l'application POSTMAN sur votre ordinateur.

Code	Fastpi_app.py
Requête POSTMAN	• Ouvrez Postman. • Créez une nouvelle requête `POST`. • Entrez l'URL : `http://127.0.0.1:8989/predict_audio` • Sous l'onglet "Body", sélectionnez "form-data". • Ajoutez un champ avec le nom `file`, et définissez-le comme `File`. Ensuite, chargez un fichier audio pour l'upload. • Cliquez enfin sur le bouton send
Résultat type attendu en cas de succès ***Exemple à partir du fichier*** *alexa_lea_audio_1_train.wav* ***dans la dataset***	{ "transcription": ------------------ "sentiment": "positif | négatif" }