Année universitaire
2024/2025
Filière
Licence d'excellence en Intelligence artificielle

FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

Speech-to-Text par Modèle Préentraîné : Pipeline de Prétraitement et Analyse des Résultats

Encadré par :
PR. BEN LAHMAR El Habib
PR. ELFAKIR Zakaria

Réalisé par :
ASSOULI Fatima zahra
BOUSSAID Salama
CHAABAN Malika

Table des matières :

Table de figure :

I. Présentation du Projet :

1. Contexte du projet :

La transformation de la parole en texte, aussi connue sous le nom de Reconnaissance Automatique de la Parole (STT), représente actuellement une technologie essentielle dans le secteur de l'intelligence artificielle. Elle est capable de convertir une voix en texte écrit, simplifiant l'accès à l'information, la communication homme-machine ou encore la conversion de contenu audio pour plusieurs usages : assistants vocaux, sous-titrage automatique, aides à la communication pour les personnes ayant des problèmes d'audition, systèmes de dictée et bien plus encore.

Historiquement, les systèmes de reconnaissance vocale se basaient sur des modèles acoustiques et linguistiques compliqués, exigeant une compétence approfondie et une personnalisation spécifique aux environnements acoustiques ou linguistiques. L'apparition de modèles de deep learning, ainsi que de modèles pré-entraînés tels que Whisper, a marqué une avancée significative dans le domaine de la reconnaissance vocale. Ces modèles ont la capacité de traiter une grande variété de données audio, d'être résistants aux bruits environnants et de gérer un large éventail d'accents et de langues sans nécessiter d'ajustements manuels.
Ce projet s'inscrit donc dans cette mouvance, en tirant parti de la force du modèle Whisper d'OpenAI pour réaliser la transcription automatique des discours en anglais, avec une attention spéciale portée à la gestion des accents régionaux et étrangers.

2. Objectifs du projet :

L'objectif premier est d'examiner l'efficacité et la solidité du modèle Whisper sur un ensemble de données illustratif de la variété des accents anglais, en se basant sur le Speech Accent Archive, une base de données prestigieuse qui rassemble des enregistrements d'orateurs natifs anglophones issus de diverses régions et nations.
Plus précisément, le projet a pour objectif de :

- **Effectuer un prétraitement efficace des fichiers audios provenant** du corpus, tout en considérant les particularités acoustiques et la variété des accents.

- **Étudier de manière acoustique ces enregistrements** en observant diverses représentations sonores : formes d'onde, spectrogrammes, mel-spectrogrammes, dans le but de mieux saisir les différences entre les accents.

- **Mettre en place une chaîne complète de transcription automatique,** allant de l'audio brut au texte, en se servant du modèle Whisper accessible via l'interface Hugging Face.

- **Juger la qualité des transcriptions** de manière quantitative en calculant le taux d'erreur de mots (Word Error Rate, WER), une mesure standard qui évalue l'écart entre la transcription automatique et celle de référence.

- **Étudier l'effet des variations d'accent** sur les résultats de transcription, en repérant les situations où le modèle performe bien ou au contraire éprouve des problèmes.

- **Étudier les restrictions et les contraintes des modèles existants de la STT** dans un contexte de diversité linguistique et phonétique.

- **Offrir une séquence reproductible** qui comprend le prétraitement audio, l'extraction de caractéristiques, la transcription et l'évaluation.

3. Présentation du modèle utilisé : Whisper :

Le modèle Whisper, développé par OpenAI, est un système avancé de reconnaissance vocale automatique (ASR) reposant sur une architecture Transformer. Il a été entraîné sur un corpus exceptionnellement vaste, comprenant plus de 680 000 heures d'enregistrements audio issus de sources hétérogènes telles que des podcasts, vidéos YouTube, conversations informelles, interviews, conférences et autres contenus disponibles sur le web.

Whisper a été élaboré en tant que modèle multitâche et multilingue. Il a la capacité d'identifier et de transcrire plus de 90 langues distinctes, avec une performance remarquable même pour les langues moins courantes. Au-delà de la simple transcription, il est également capable d'exécuter diverses tâches comme l'identification automatique de la langue parlée et, dans certains scénarios, la conversion de l'audio en anglais. C'est cette adaptabilité qui en fait un instrument très exhaustif pour la gestion de la parole.
L'un des principaux avantages de Whisper est sa capacité à fonctionner efficacement dans des conditions réelles : il gère parfaitement le bruit de fond, les accents régionaux et les variations de prononciation, ce qui est crucial pour les applications en milieu non contrôlé (service client, transcription de réunions, accessibilité, etc.).

Dans le cadre de ce projet, le modèle Whisper est utilisé dans sa version pré-entraînée, disponible via la bibliothèque Transformers de Hugging Face. Cela permet une intégration rapide et flexible dans un pipeline de traitement, sans nécessiter d'entraînement spécifique. L'utilisation de cette version facilite également l'expérimentation, le prototypage rapide et l'évaluation des performances du modèle sur des jeux de données spécifiques au domaine d'étude.

Architecture du modèle Whisper :

- Architecture Transformer encodeur-décodeur, optimisée pour la transcription et la traduction automatique de la parole.
- Spectrogramme log-Mel en entrée, permettant une représentation fine du signal audio avant traitement par l'encodeur.
- Décodeur autoregressif, générant le texte mot par mot, en se basant à la fois sur l'audio et les séquences précédentes.
- Jetons de tâche (task tokens) intégrés pour indiquer explicitement l'objectif : transcription, détection de langue ou traduction.
- Conception multilingue et multitâche, assurant une grande polyvalence sur plus de 90 langues.
- Modèles de tailles variées (Tiny, Base, Small, Medium, Large) permettant de s'adapter aux contraintes de performance et de calcul.

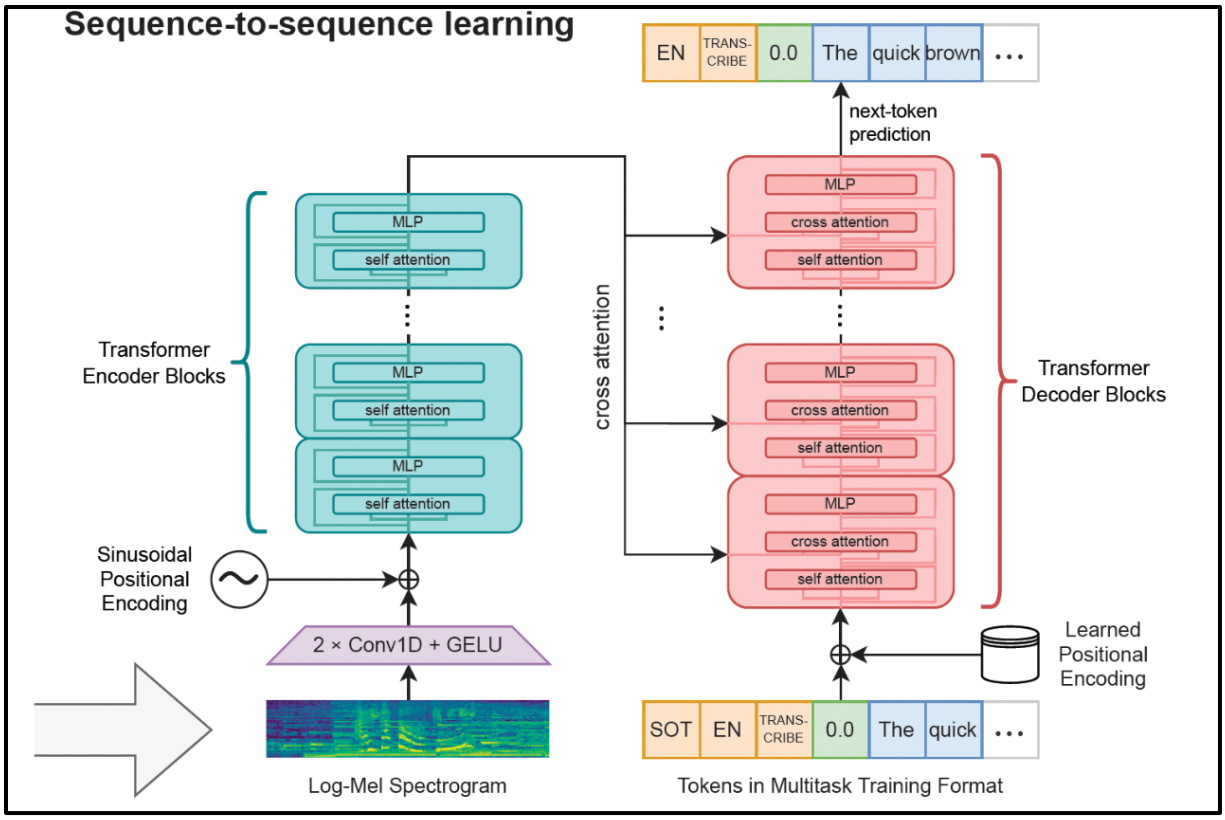


Figure 1 : Architecture du modèle

4. Environnement et outils techniques utilisés :

Le projet a été développé en Python, au sein d'un environnement interactif Jupyter Notebook, facilitant le prototypage, la visualisation et l'expérimentation.

Les principaux outils utilisés sont :

- **Transformers (Hugging Face)** : pour le chargement du modèle Whisper et la gestion du pipeline de transcription.
- **Torchaudio** et **Librosa** : bibliothèques spécialisées pour la manipulation, le prétraitement, et l'analyse des fichiers audio, notamment pour le rééchantillonnage, le découpage, et la génération de spectrogrammes.
- **Jiwer** : une bibliothèque dédiée au calcul du Word Error Rate (WER), permettant une évaluation précise de la qualité des transcriptions.
- **Matplotlib** et **Seaborn** : pour la visualisation des formes d'onde et des spectrogrammes, afin d'analyser visuellement les caractéristiques acoustiques.

Cette sélection d'outils offre un compromis optimal entre puissance, flexibilité, et simplicité d'utilisation, permettant de construire une solution complète de reconnaissance vocale, allant de la préparation des données à l'évaluation des résultats.

II. Méthodologies :

Chargement et prétraitement des données :

1. Importation des données:

Le **Speech Accent Archive** est un corpus linguistique conçu pour illustrer la diversité des accents anglais à travers le monde. Chaque participant, locuteur natif ou non, lit **le même paragraphe en anglais**, ce qui permet de comparer facilement les variations d'accent.

Le jeu de données comprend :

- **Fichiers audio (.mp3)** : chaque fichier correspond à un locuteur lisant le paragraphe standardisé.
- **Fichier texte** : contient le paragraphe de référence lu par tous les locuteurs.

Les enregistrements couvrent **plus de 100 nationalités**, offrant une grande richesse phonétique. Ce corpus est idéal pour tester la **robustesse du modèle Whisper** face aux accents variés et aux conditions de parole réelles.

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("rtatman/speech-accent-archive")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/rtatman/speech-accent-archive?dataset_version_number=2...
100%|██████████| 865M/865M [00:11<00:00, 76.4MB/s]
Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/rtatman/speech-accent-archive/versions/2
```

Figure 2 : Importation de la base de données

2. Termes clés concernant les fichiers audio :

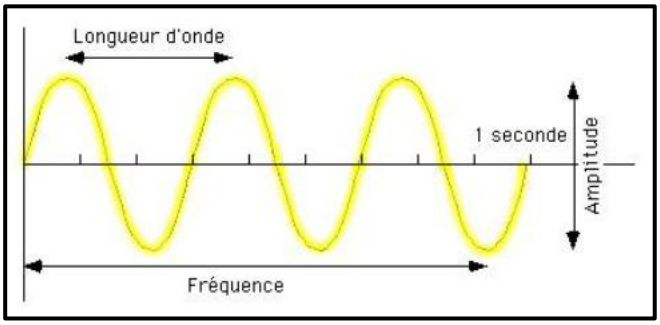
Fréquence:

- La fréquence d'un audio correspond au nombre de vibrations (oscillations) d'une onde sonore par seconde.
- 20 – 250 Hz Basses (sons graves)
- 250 – 2000 Hz Médiums (paroles humaines)
- 2000 – 20000 Hz Aigus (sons perçants)
- L'oreille humaine entend en général de 20 Hz à 20 000 Hz.
- 1000 Hz (1 kHz) = 1000 vibrations par seconde.

Intensité (db/ power):

- L'intensité (ou amplitude) d'un audio mesure combien d'énergie transporte l'onde sonore.
- Elle est liée à la hauteur des vagues de l'onde sonore.

- Plus l'onde est « haute », plus le son est fort.



Sampling rate:

- Le sampling rate (ou fréquence d'échantillonnage) est le nombre d'échantillons que l'on prend par seconde pour représenter un son analogique sous forme numérique.

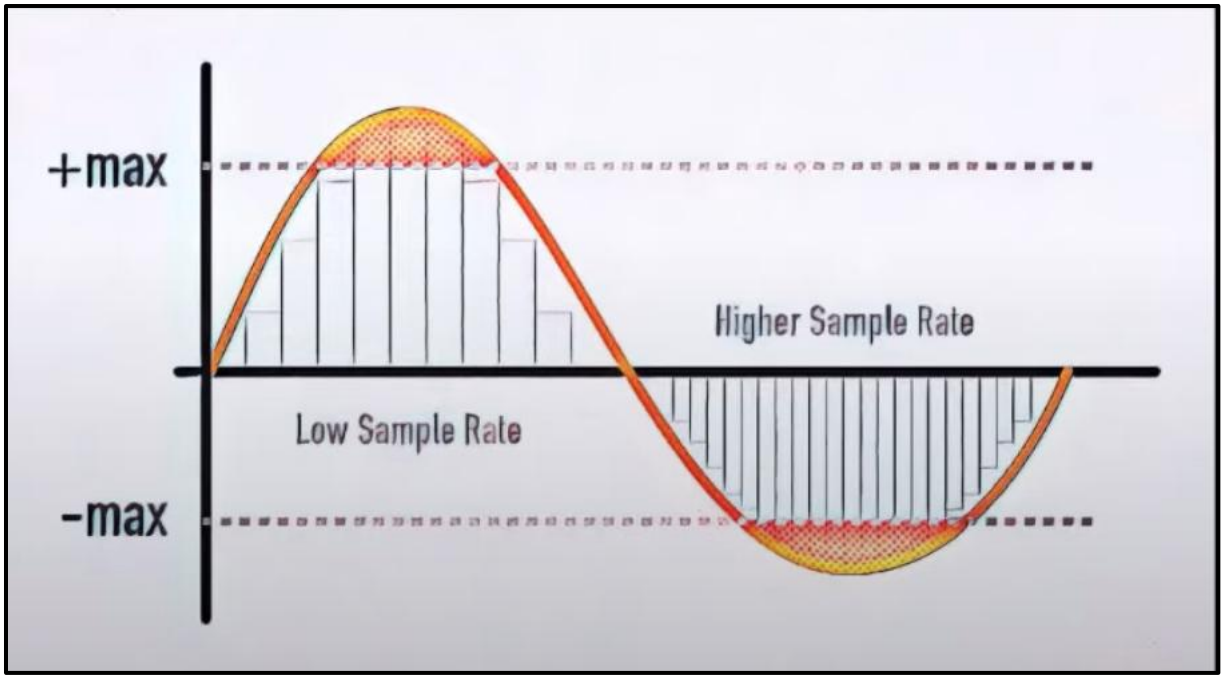


Figure 3 : Sampling rate

3. Prétraitement des données audio :

On commence par récupérer la liste des fichiers audio.

```
audio_files = glob('/content/drive/MyDrive/English-accent-dataset/recordings/recordings/*.mp3')
```

```
ipd.Audio(audio_files[0])
```

0:00

Figure 4 : Récupération des fichiers audios

Puis on charge le fichier audio avec librosa :

- y est un tableau contenant les amplitudes du signal audio.
- sr est la fréquence d'échantillonnage (sampling rate)

```
y, sr= librosa.load(audio_files[0]) # y est un array qui contient les amplitudes du fichier audio
print(f'y :{y[:10]}')
print(f'sr : {sr}')
```

```
y :[ 0.00323694 -0.00356178 -0.00168397  0.00269967 -0.00282407  0.00654717
  0.00021672  0.00351303 -0.00146281  0.00273623]
sr : 22050
```

```
pd.Series(y)
```

	0
0	0.003237
1	-0.003562
2	-0.001684
3	0.002700
4	-0.002824
...	...
492519	-0.000015
492520	-0.001528
492521	-0.000951
492522	-0.004807
492523	-0.008012

492524 rows × 1 columns

Figure 5 : Chargement du fichier audio

Visualisation de l'évolution du signal :

```
pd.Series(y).plot(figsize=(10,5))
```

```
<Axes: >
```

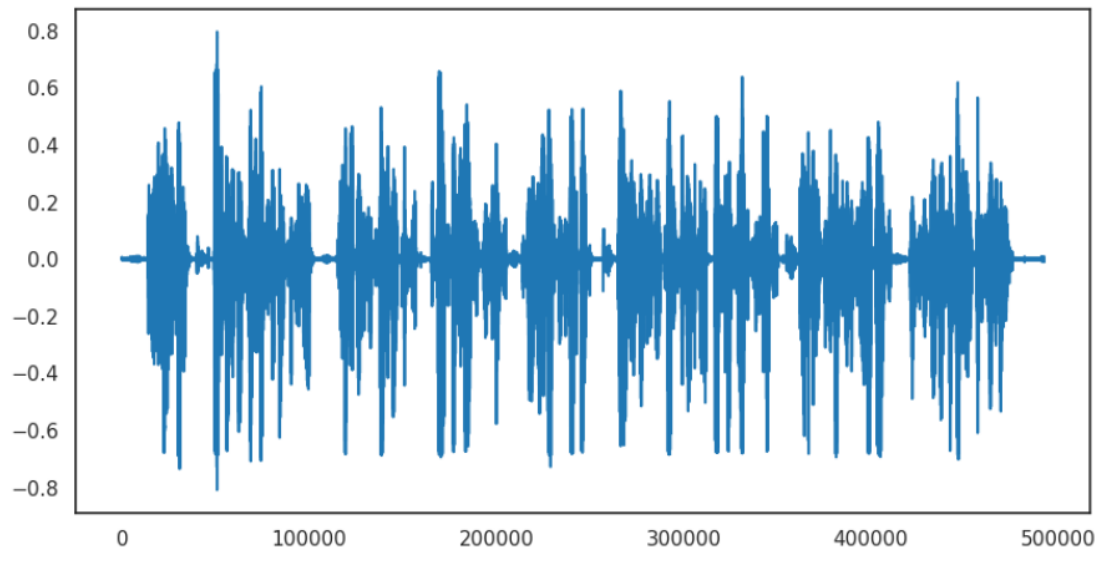


Figure 6 : visualisation d'évolution du signal

Nettoyage et découpage du signal :

On élimine les silences ou les segments peu informatifs à l'aide d'une fonction de **trim** de librosa :

```
y_trimmed,_=librosa.effects.trim(y, top_db=5)
y_trimmed
```

```
array([0.01013541, 0.02473876, 0.04247479, ..., 0.01330926, 0.0154721 ,
       0.01479802], dtype=float32)
```

```
pd.Series(y_trimmed).plot(figsize=(10,5))
```

```
<Axes: >
```

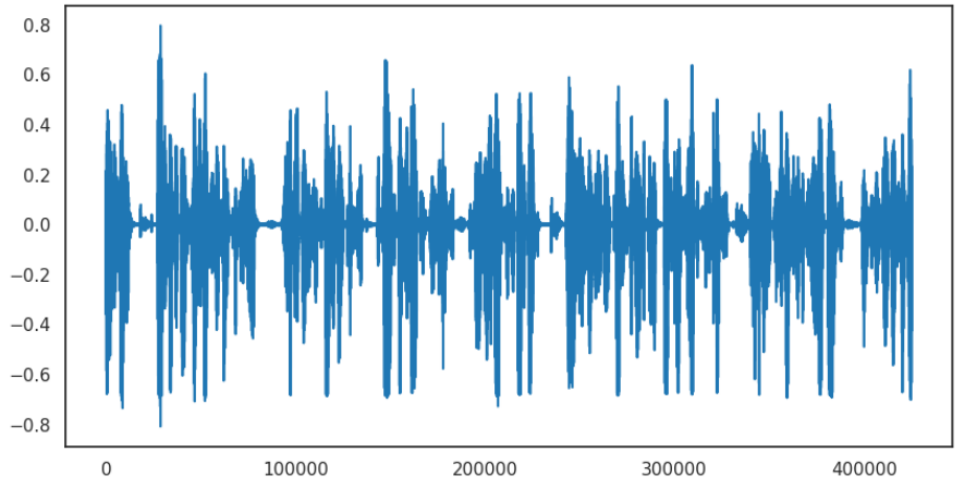


Figure 7 : Nettoyage et découpage du signal

5. Représentation spectrale :

Spectrogramme : représentation visuelle en deux dimensions de l'évolution des fréquences au cours du temps. Il est généré grâce à la **Short-Time Fourier Transform (STFT)** qui découpe le signal en fenêtres temporelles , puis applique une transformation de Fourier sur chacune :

```
D=librosa.stft(y)
S_db =librosa.amplitude_to_db(np.abs(D), ref=np.max)
S_db.shape
```

```
(1025, 962)
```

```
fig, ax= plt.subplots(figsize=(10,5))
img=librosa.display.specshow(S_db,
                             x_axis='time',
                             y_axis='log',
                             ax=ax)
ax.set_title('Spectrogramme', fontsize=20)
fig.colorbar(img, ax=ax, format=f'%0.2f')
plt.show()
```

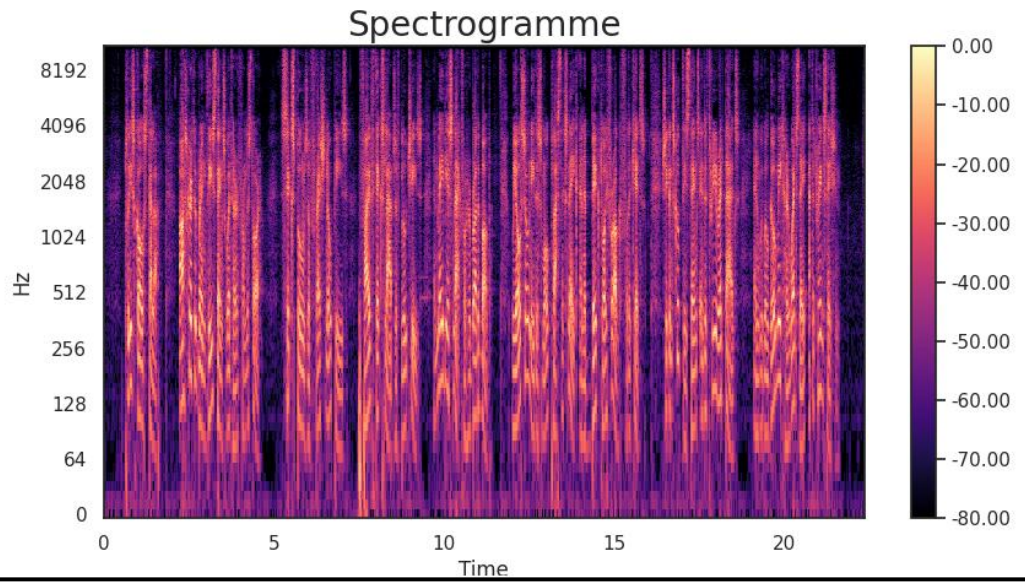


Figure 8 : Spectrogramme

Mel Spectrogramme :
Il s'agit d'une version du spectrogramme adaptée à la perception humaine des sons grâce à l'échelle Mel, qui modifie la manière dont les fréquences sont représentées.

- L'échelle **Mel** convertit les fréquences (Hz) en une échelle plus proche de notre perception auditive.
    - **Graves (20–500 Hz)** : plus détaillés, correspondant à des sons comme un tambour ou une voix profonde.
    - **Aigus (2000–20000 Hz)** : plus compressés, typiques de sons comme un sifflet ou la lettre "s".
- Contrairement à une échelle linéaire où les fréquences sont uniformément espacées, l'échelle Mel reflète le fait que l'oreille humaine est plus sensible aux sons graves.
- Le **Mel spectrogramme** est donc une représentation transformée :
    - **X** : le temps
    - **Y** : les fréquences perçues (en Mels)
    - **Couleur** : l'intensité sonore à chaque instant

```python
S=librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128)
S_db_mel =librosa.amplitude_to_db(S, ref=np.max)

fig, ax= plt.subplots(figsize=(15,5))
img=librosa.display.specshow(S_db_mel,
                             x_axis='time',
                             y_axis='log',
                             ax=ax)
ax.set_title('Mel Spectrogramme', fontsize=20)
fig.colorbar(img, ax=ax, format=f'%0.2f')
plt.show()
```

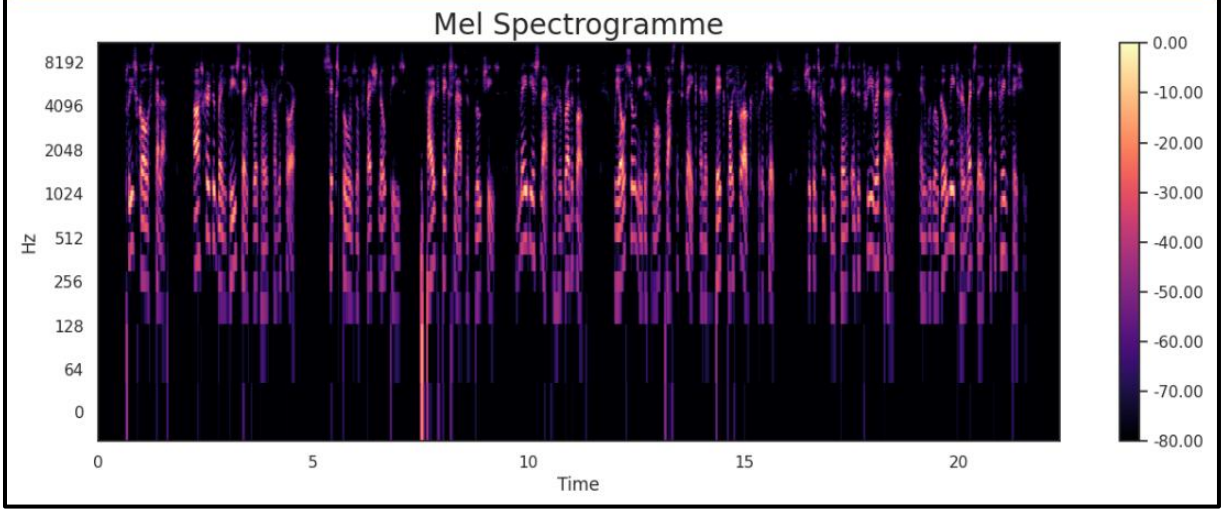


Figure 9 : Mel Spectrogramme

6. Prétraitement des données à Whisper :

Le modèle **Whisper** requiert des entrées audio normalisées à **16 kHz**. Ainsi, les étapes suivantes sont nécessaires :

1. Chargement de l'audio avec torchaudio
2. Vérification de la fréquence d'échantillonnage et rééchantillonnage si nécessaire
3. Conversion en tableau NumPy pour traitement

```python
audio_path = audio_files[0]
speech_array, sampling_rate = torchaudio.load(audio_path)

# Resample to 16kHz if needed
if sampling_rate != 16000:
    resampler = torchaudio.transforms.Resample(orig_freq=sampling_rate, new_freq=16000)
    speech_array = resampler(speech_array)

# Convert to 1D numpy array
import numpy as np
speech = speech_array[0].numpy()
```

```python
from transformers import WhisperProcessor, WhisperForConditionalGeneration

model_name = "openai/whisper-small"
processor = WhisperProcessor.from_pretrained(model_name)
model = WhisperForConditionalGeneration.from_pretrained(model_name)
```

Figure 10 : Prétraitement des données pour Whisper

```python
inputs = processor(speech, sampling_rate=16000, return_tensors="pt")
```

Figure 11 : Conversion des données en tableau

7. Chargement du texte de référence :

Le paragraphe standard, lu par tous les locuteurs du corpus, est stocké dans un fichier texte. Pour pouvoir l'utiliser comme référence lors de l'évaluation des transcriptions générées par Whisper, il est nécessaire de le nettoyer en supprimant la ponctuation et les caractères spéciaux. Cela permet d'assurer une comparaison plus fiable avec le texte transcrit automatiquement.

```
filename= '/content/drive/MyDrive/English-accent-dataset/reading-passage.txt'
with open(filename, 'r') as f:
  lines=f.readlines()
phrase=[]
for line in lines:
  line=line.strip().split()
  line=" ".join(line)
  phrase.append(line)
phrase=phrase[0].replace(',',"").replace('.',"").replace(':',"").replace("'","")
phrase
```

Figure 12 : Chargement du texte référence

8. Transcription avec Whisper :

Whisper est un modèle de transcription automatique de la parole développé par **OpenAI**. Il repose sur une architecture **encoder-decoder de type Transformer** et a été entraîné sur un vaste corpus multilingue contenant des centaines de milliers d'heures d'audio.

Whisper est conçu pour être **robuste aux accents**, aux bruits de fond et aux variations de qualité audio. Il peut effectuer plusieurs tâches, telles que la **transcription**, la **traduction**, la **détection de langue**, ou encore la **diarisation** (reconnaissance des locuteurs).

Étapes de transcription avec Whisper :

- **Génération des prédictions (IDs de tokens) :**

```
# Generate predicted token IDs
predicted_ids = model.generate(inputs.input_features)
```

Figure 13 : Géneration des IDs de Tokens

- **Décodage en texte brut :**

```
transcription = processor.batch_decode(predicted_ids, skip_special_tokens=True)[0]
print("Transcription:", transcription)
```

Figure 14 : Transcription

- **Nettoyage de la transcription :**

```
transcription=transcription.replace(',',"").replace('.',"").replace(':',"").replace("'","").strip()
transcription
```

Figure 15 : Nettoyage de la transcription

III. Résultats et évaluation des performances :

L'évaluation d'un système de transcription automatique comme **Whisper** est une étape clé pour juger de sa fiabilité et de sa robustesse, en particulier dans un contexte multilingue et accentué. Deux métriques principales ont été utilisées dans notre étude : la **Word Error Rate (WER)**, qui mesure les erreurs au niveau des mots, et la **Character Error Rate (CER)**, qui évalue la précision au niveau des caractères. L'utilisation conjointe de ces deux indicateurs permet d'obtenir une vision plus fine de la qualité des transcriptions générées.

1. Définition des métriques utilisées :

- Le **WER (Word Error Rate)** est une mesure quantitative de l'exactitude d'une transcription. Elle indique le **taux d'erreurs** dans la transcription générée par rapport à une **transcription de référence** (le texte exact censé être prononcé).

Elle est calculée comme suit :

$$\mathbf{WER} = \frac{S+D+I}{N}$$

Figure 16 : Formule du WER

- **S (Substitutions)** : un mot est remplacé par un autre incorrectement.
- **D (Suppressions)** : un mot attendu est absent dans la transcription.
- **I (Insertions)** : un mot est ajouté sans raison valable.
- **N** : nombre total de mots dans le texte de référence.

Plus le **WER est faible**, meilleure est la transcription. Un **WER = 0.0** signifie une correspondance parfaite.

- **Character Error Rate (CER)** est une métrique plus fine, qui mesure les erreurs au **niveau des caractères**. Elle est particulièrement utile pour détecter de petites déviations, comme une lettre manquante, une faute de frappe, ou une erreur de segmentation. Elle est calculée de manière similaire au WER, mais en se basant sur les caractères plutôt que les mots :

$$CER = \frac{S + D + I}{N}$$

Figure 17 : Formule CER

2. Mise en œuvre dans notre expérimentation :

On a évalué les transcriptions générées par Whisper à partir d'un sous-ensemble de 210 fichiers audio du corpus Speech Accent Archive, tous contenant la même phrase lue par des locuteurs de différentes origines. Pour chaque audio :

- Le signal a été chargé et, si nécessaire, rééchantillonné à 16 kHz pour correspondre aux exigences du modèle.
- Le texte généré a été nettoyé (suppression de la ponctuation et normalisation).
- Les métriques WER et CER ont été calculées à l'aide de la bibliothèque jiwer.
- Les résultats ont été visualisés graphiquement, pour évaluer les variations des performances d'un locuteur à l'autre.

```
from jiwer import wer
error = wer(phrase,transcription)
print(f"Predicted: {transcription}")
print(f'WER: {error}')
```

```
Predicted: Please call Stella Ask her to bring these things with her from the store Six spoons of fresh snow peas five thick slabs of blue cheese and ma
ybe a snack for her brother Bob We also need a small plastic snake and a big toy frog for the kids She can scoop these things into three red bags and we
will go meet her Wednesday at the train station
WER: 0.0
```

Figure 18 : Calcul de WER

```
import jiwer
cer = jiwer.cer(transcription,phrase)

print(f"CER: {cer:.2%}")
```

```
CER: 0.00%
```

Figure 20 : Calcul du CER

```
print(transcription)
print(phrase)
```

```
Please call Stella Ask her to bring these things with her from the store Six spoons of fresh snow peas five thick slabs of blue cheese and maybe a snack
for her brother Bob We also need a small plastic snake and a big toy frog for the kids She can scoop these things into three red bags and we will go mee
t her Wednesday at the train station
Please call Stella Ask her to bring these things with her from the store Six spoons of fresh snow peas five thick slabs of blue cheese and maybe a snack
for her brother Bob We also need a small plastic snake and a big toy frog for the kids She can scoop these things into three red bags and we will go mee
t her Wednesday at the train station
```

Figure 19 : Résultat obtenu

3. Analyse des résultats :

Ce morceau de code effectue l'évaluation automatique des transcriptions générées par le modèle Whisper sur un ensemble d'enregistrements audio, en utilisant deux métriques : WER (Word Error Rate) et CER (Character Error Rate).

```
wer_scores = []
cer_scores = []
input =[]
for audio_path in audio_files[:210]:
    speech_array, sampling_rate = torchaudio.load(audio_path)
    if sampling_rate != 16000:
      resampler = torchaudio.transforms.Resample(orig_freq=sampling_rate, new_freq=16000)
      speech_array = resampler(speech_array)
    speech = speech_array[0].numpy()

    inputs = processor(speech, sampling_rate=16000, return_tensors="pt")
    input.append(inputs)
```

```
for inputs in input:

    predicted_ids = model.generate(inputs.input_features)
    transcription = processor.batch_decode(predicted_ids, skip_special_tokens=True)[0]
    print(transcription)
    transcription=transcription.replace(',',"").replace('.',"").replace(':',"").replace("'","").strip()

    error = wer(phrase,transcription)
    wer_scores.append(error)
    cer = jiwer.cer(transcription,phrase)
    cer_scores.append(cer)
print(wer_scores[:10])
print(cer_scores[:10])
```

- Pour chaque fichier audio, on obtient une transcription automatique.
- On compare cette transcription au texte de référence, en mesurant les erreurs **aux niveaux des mots (WER)** et **des caractères (CER)**.
- On stocke ces résultats pour pouvoir ensuite les **analyser ou visualiser**.

Les graphiques ci-dessous illustrent l'évolution du **WER** et du **CER** pour chaque audio :

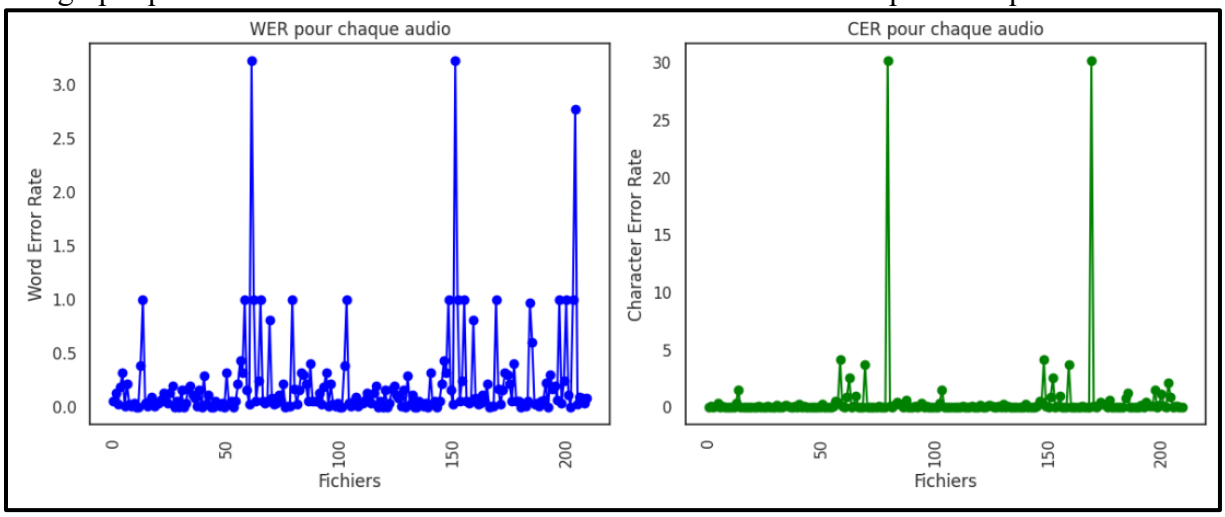


Figure 21 : L'évolution des métriques pour chaque audio

- WER:

Score	Interprétation	Exemple
0.00	Parfait (aucune erreur)	Transcription exacte
0.01 - 0.10	Excellent (quasiment parfait)	Quelques petites erreurs
0.11 - 0.20	Bon	Peut contenir quelques fautes mais globalement compréhensible
0.21 - 0.40	Moyen	Erreurs notables, mais transcription utile
0.41 - 0.60	Faible	Beaucoup d'erreurs, transcription peu fiable
> 0.60	Mauvais	Résultat quasi inutilisable

-
- CER:
  - CER < 10% = excellent
  - CER < 20% = acceptable
  - CER >= 40% = problématique

Figure 22 : L'interprétation des graphes

L'analyse croisée du WER et du CER renforce notre constat initial : le modèle Whisper s'avère hautement performant, même lorsqu'il est confronté à une diversité importante d'accents. De plus, la qualité des résultats témoigne aussi de la pertinence du pipeline de traitement audio mis en place (prétraitement, resampling, nettoyage), et de la robustesse du modèle face aux défis typiques de la transcription multilingue.

Conclusion :

Au terme de cette étude, nous avons pu mettre en lumière les capacités remarquables du modèle Whisper, développé par OpenAI, dans le domaine de la transcription automatique de la parole (Speech-to-Text). En nous appuyant sur le corpus Speech Accent Archive, reconnu pour sa richesse phonétique et la diversité des accents représentés, nous avons soumis le modèle à un véritable stress-test linguistique. L'objectif principal était d'évaluer la robustesse du modèle face à la variabilité des accents de l'anglais parlé, y compris chez des locuteurs non natifs.

Un pipeline complet a été mis en place : prétraitement audio (nettoyage, rééchantillonnage, suppression des silences), visualisation acoustique (formes d'onde, spectrogrammes, mel-spectrogrammes), transcription avec Whisper, et enfin évaluation des performances à l'aide de deux métriques complémentaires : le Word Error Rate (WER) et le Character Error Rate (CER), grâce à la bibliothèque jiwer.

Les résultats que l'on a obtenus sont particulièrement notables. Nous avons noté des résultats de WER atteignant 0.0, ce qui indique une transcription impeccable, mais également des CER extrêmement bas (occasionnellement inférieurs à 5%), attestant de la précision des transcriptions jusque dans les nuances de caractères. L'association de ces deux indicateurs permet une analyse détaillée : tandis que le WER offre une perspective globale sur la construction des phrases, le CER fournit un niveau de détail plus pointu, essentiel pour identifier les fautes discrètes, en particulier concernant les mots similaires ou les noms propres.

Ces performances mettent en évidence non seulement les capacités techniques avancées de Whisper (entraînement multilingue à grande échelle, robustesse face aux accents et bruits légers), mais aussi l'efficacité du pipeline audio mis en œuvre dans ce projet. Le faible taux d'erreur, y compris en présence d'accents fortement marqués, témoigne de la maturité du modèle et de la qualité du traitement préalable.

Cette recherche met finalement en lumière l'importance d'utiliser des modèles préformés robustes tels que Whisper-small pour des travaux sophistiqués, sans avoir besoin d'une étape de réentraînement, tout en insistant sur le fait que la qualité du traitement audio initial est cruciale pour obtenir des résultats fiables.
En définitive, ce projet prouve qu'il est maintenant possible d'avoir des solutions robustes, accessibles et efficaces pour la transcription automatique de la parole. Des modèles tels que Whisper permettent de rendre la parole numérique plus accessible à un large public, dans divers contextes multilingues et inclusifs.