Mastères de recherche:

Information System techniques (IST)

Projet Reconnaissance Vocale

Travaillé par :

Nourhene Belhadj

Niveau :

2éme année mastère IST

Année Scolaire :

2024/2025

Sommaire :

1. Présentation du projet :

Mon projet de reconnaissance vocale vise à développer un système capable de convertir la parole en texte avec précision, en utilisant des techniques avancées de traitement du signal et d'apprentissage automatique. Ce système pourra être utilisé dans divers domaines, tels que les assistants vocaux, les sous-titres automatiques et l'accessibilité pour les personnes en situation de handicap**.**

2. Configuration d'un environnement de développement

- Python 3.10.5 : ***$ pip install python 3.10.5***
- PyAudio 0.2.14 : ***$ pip install pyaudio***
- PocketSphinx : ***$ pip install pocketsphinx***
- Google API Client Library for Python : ***$ pip install google-api-python-client***
- Whisper : ***$ pip install git+https://github.com/openai/whisper.git***
- openai : ***$ pip install openai***
- ffmpeg (required for handling audio formats like WAV, MP3) :
  - ***Télécharger package ffmpeg en formant .zip fichier de site*** **https://ffmpeg.org/download.html**

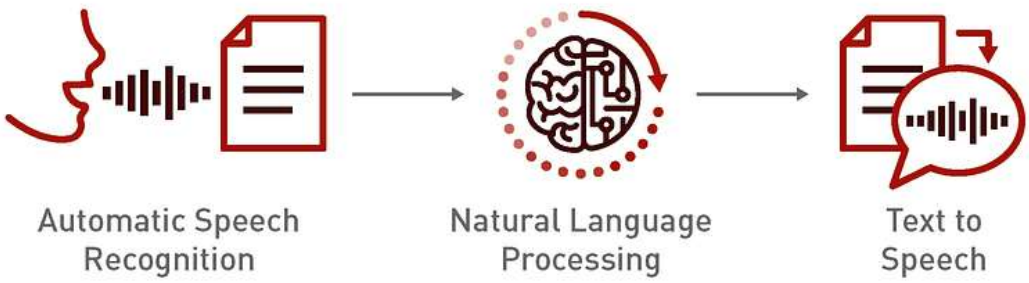


Figure 1 : La solution de la reconnaissance

3. Architecture de la solution implémentée :

➢ Capture du signal audio :

- Le **microphone** est utilisé comme source d'entrée pour capturer la voix de l'utilisateur.
- L'objet Microphone ***(de la bibliothèque speech_recognition)*** permet de se connecter au microphone et de lire les données audios en temps réel.

➢ Traitement du signal audio :

- L'objet Recognizer agit comme un moteur central qui analyse et interprète les données audios pour effectuer la reconnaissance vocale.
- Le Recognizer applique ***des techniques de traitement du signal*** pour convertir les ondes sonores en texte.

➢ Pipeline de traitement :

- ***Initialisation :***
    - Un objet Recognizer est instancié pour gérer les tâches de reconnaissance.
    - L'objet Microphone est instancié pour accéder au matériel d'entrée.
- ***Lecture audio :***
    - L'audio est capturé via l'objet Microphone dans un contexte (with statement).
- ***Conversion audio-texte :***
    - L'audio capturé est envoyé au Recognizer, qui peut utiliser des services comme Google Speech API pour transcrire la parole en texte.
    -

➢ **Gestion des erreurs :**

- Les exceptions comme UnknownValueError (audio incompréhensible) ou RequestError (problèmes réseau) sont gérées pour garantir la robustesse de l'application.

4. Implémentation du code de reconnaissance vocal

➢ **Libraires utilisés pour traitement de données :**

1. **audioop :** fonctions permettant de manipuler des données audios brutes (par exemple, réglage du volume, détection des crêtes et conversion des formats audio). Généralement utilisé pour le traitement audio de bas niveau.
2. **Base64 :** Fournit des fonctions pour encoder et décoder des données en Base64, un format d'encodage binaire basé sur le texte souvent utilisé pour la transmission de données dans les applications Web ou les pièces jointes aux courriels.
   Cas d'utilisation : Utile pour les tâches impliquant des structures de données avec des fonctionnalités avancées.
3. **hashlib :** offre des fonctions de hachage cryptographique telles que SHA-1, SHA-256 et MD5 pour le hachage sécurisé des données.

Cas d'utilisation : Couramment utilisé pour les contrôles d'intégrité des données et le hachage de mots de passe.

4. **hmac :** implémente la fonction HMAC (Keyed-Hashing for Message Authentication) pour vérifier l'intégrité et l'authenticité des données à l'aide d'une clé secrète.

Cas d'utilisation : Utile pour la communication sécurisée dans les API et les services web.

5. **io :** Fournir des outils pour gérer les flux d'E/S (entrée/sortie), tels que la lecture/écriture de fichiers ou de flux binaires/textuels en mémoire.

Cas d'utilisation : Prise en charge des opérations avancées sur les fichiers et de la manipulation des flux.

6. **Wave :** permet de lire et d'écrire des fichiers audio WAV (Waveform Audio File Format).

Cas d'utilisation : Utilisé pour traiter les données audios au format WAV

7. **uuid :** génère des identificateurs universels uniques (UUID) pour identifier de manière unique des objets ou des données.

Cas d'utilisation : Couramment utilisé pour les clés de base de données, les jetons de session et les systèmes distribués.

8. **Time :** Fournit des fonctions permettant de travailler avec le temps, telles que sleep(), time() et ctime().

Cas d'utilisation : Utilisé pour mesurer le temps d'exécution, introduire des délais ou travailler avec des timestamps.

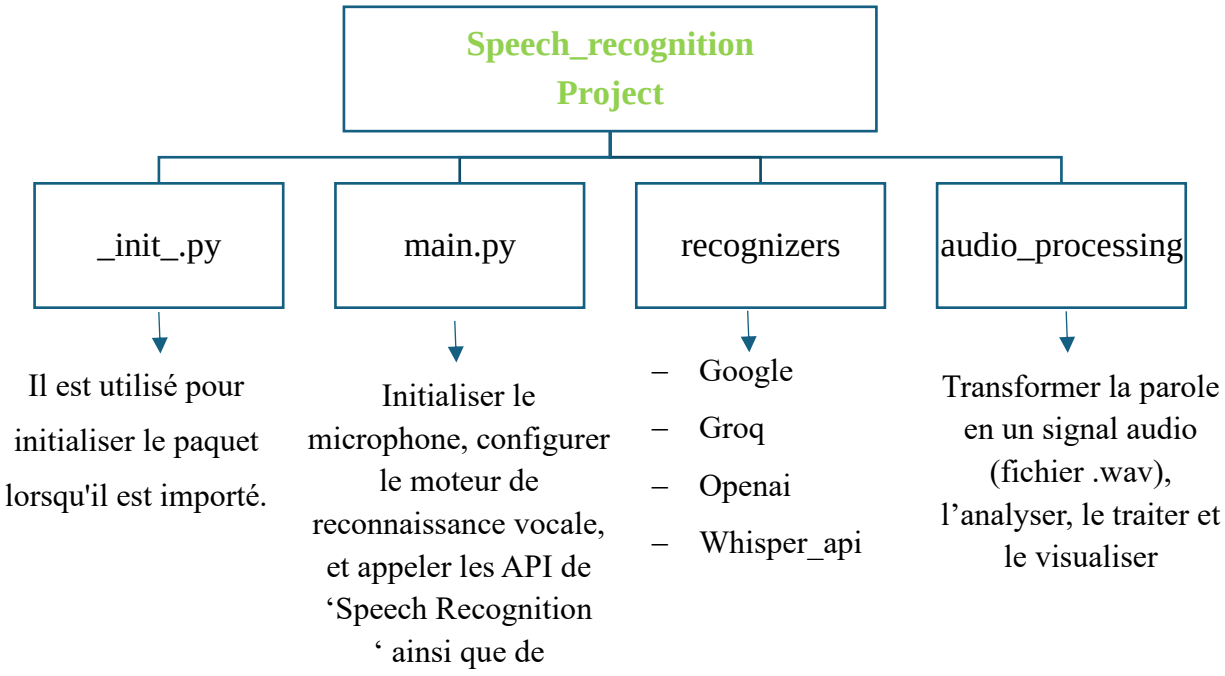
9. **Subprocess** : permet de créer de nouveaux processus, de se connecter à leurs tuyaux d'entrée/sortie/erreur et de récupérer leurs codes de retour.

Cas d'utilisation : Utilisé pour exécuter des commandes shell ou des programmes externes à partir de Python.

10. **Os** : Fournit des fonctions permettant d'interagir avec le système d'exploitation, telles que la manipulation de fichiers, les variables d'environnement et la gestion des processus.

Cas d'utilisation : Permet d'effectuer des tâches telles que la gestion des chemins d'accès aux fichiers, la navigation dans les répertoires et l'exécution des commandes de l'interpréteur de commandes.

➢ Les répertoires de la solution (directories of the solution)



➢ Différents outils de reconnaissance dans la classe « Recognizer » :

Recognizer.recognize_api = classmethod(recognize_api) :

– L'enveloppe classmethod permet à la fonction **recognize_api** d'être appelée directement sur la classe, sans nécessiter d'instance.

→ Des API différentes où **la langue** est passé comme argument

- recognize_sphinx
- recognize_google_cloud
- recognize_azure
- recognize_bing
- recognize_ibm
- recognize_whisper
- recognize_vosk

→ Le tableau des langues qui sont déclarés et peut être passés comme argument dans les méthodes « Recognizer » : https://cloud.ibm.com/docs/natural-language-understanding?locale=fr&topic=natural-language-understanding-detectable-languages&mhsrc=ibmsearch_a&mhq=langue

➢ Résultats de la solution implémentée :

- La figure suivante représente le signal audio du mot prononcé « Hello » :
- Chaque alphabet est écrit en correspondance avec une partie de signal audio.
- Le moteur de reconnaissance vocale utilisé est **Google API**.

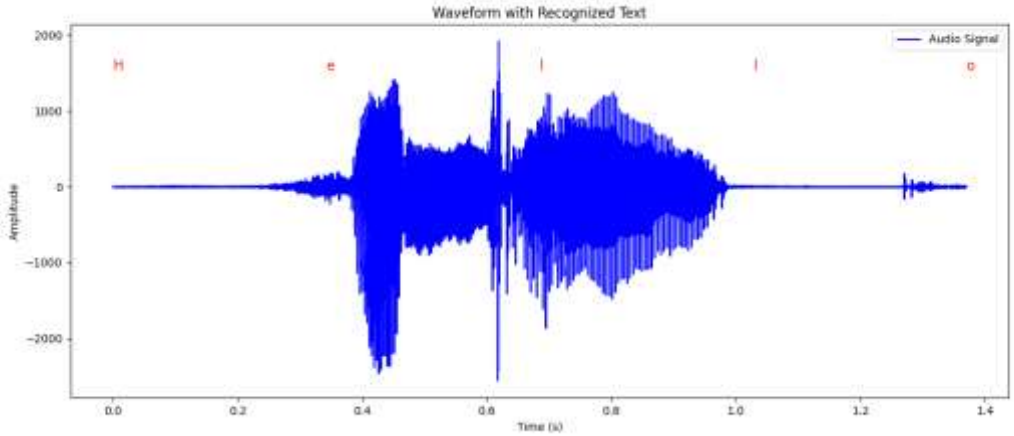


Figure 2 : Signal audio du mot « Hello » avec l'outil Google

- La 2e image représente une analyse récapitulative du signal audio.
- Elle présente les parties voisées et non voisées en établissant un taux d'énergie et un TPZ (taux de passage par zéro) dynamiques**.**

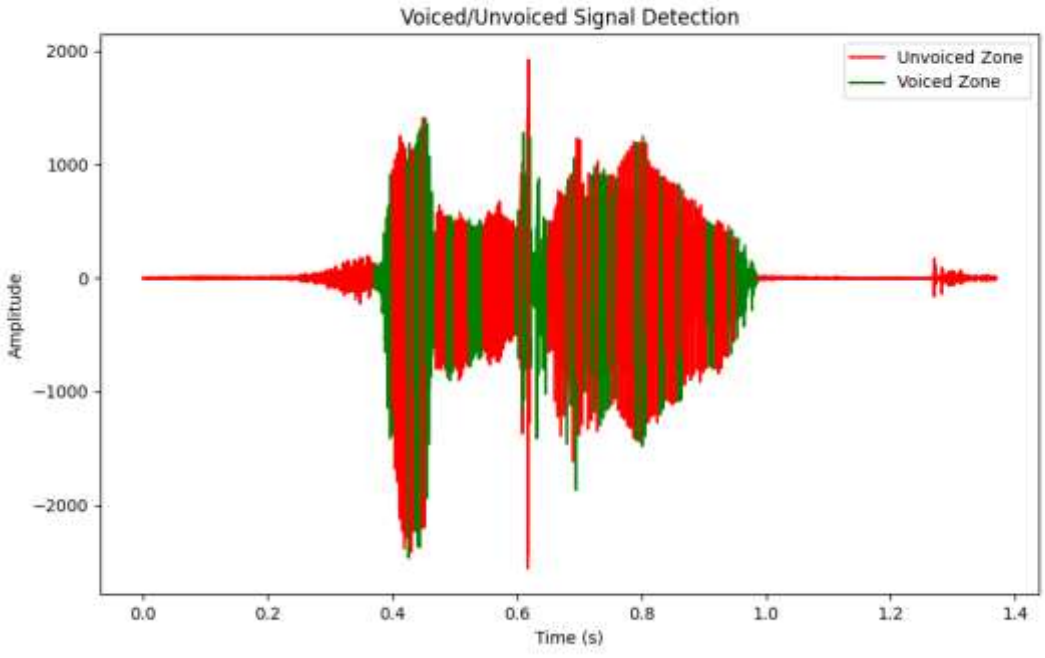


Figure 3 : Les zones voisées et non voisées du mot « Hello »

– Le moteur de reconnaissance vocale utilisé est **Whisper API :**

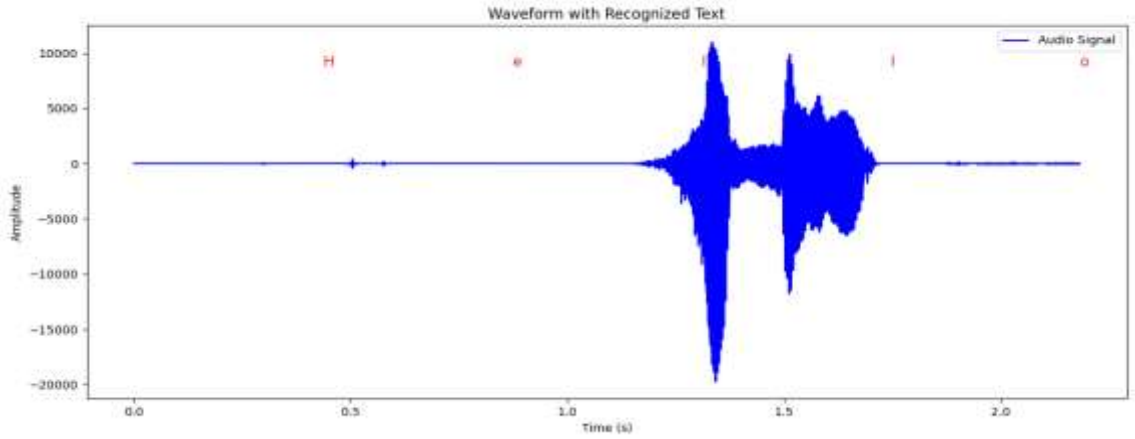


Figure 4 : Signal audio du mot « Hello » avec l'outil Whisper

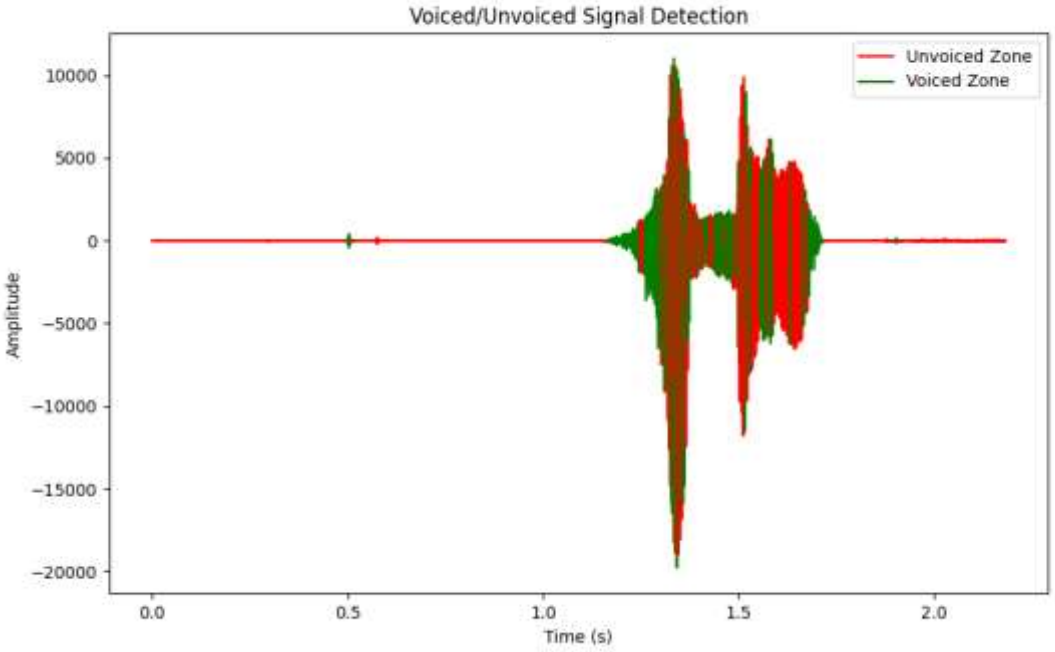


Figure 5 : Les zones voisées et non voisées du mot « Hello »

5. Conclusion :

Le projet de reconnaissance vocale s'appuie sur des outils performants et flexibles pour transformer la parole en texte de manière intuitive et efficace. Grâce à des bibliothèques telles que **SpeechRecognition**, il est possible d'intégrer facilement des fonctionnalités de reconnaissance vocale dans des applications variées.

L'approche adoptée démontre la puissance des API vocales (comme Google Speech API ou Whisper) dans la gestion de tâches complexes, tout en offrant une compatibilité avec **différentes langues** et accents.

En conclusion, ce projet met en évidence la capacité croissante des technologies modernes à rapprocher les interactions homme-machine de la communication naturelle.