Google Cloud Speech-to-Text

J'ai récemment expérimenté avec l'API Speech-to-Text de Google Cloud. Voici une fonction Python que j'ai utilisée pour effectuer la transcription.

```
import os
import json
import time
import argparse
from google.cloud import speech
from pydub import AudioSegment
import tempfile

# Répertoire de sortie fixe
OUTPUT_DIRECTORY = "assets/transcriptions"



def speech_to_text(audio_file, output_filename):
    print(f"Génération de la transcription pour : {output_filename}")
    try:
        client = speech.SpeechClient()

        # Charger le fichier audio avec pydub pour déterminer les paramètres
        audio_segment = AudioSegment.from_file(audio_file)
        sample_rate = audio_segment.frame_rate
        channels = audio_segment.channels

        # Déterminer l'encodage en fonction de l'extension du fichier
        file_extension = os.path.splitext(audio_file)[1].lower()
        if file_extension == '.mp3':
            encoding = speech.RecognitionConfig.AudioEncoding.MP3
        elif file_extension in ['.wav', '.wave']:
            encoding = speech.RecognitionConfig.AudioEncoding.LINEAR16
        elif file_extension == '.flac':
            encoding = speech.RecognitionConfig.AudioEncoding.FLAC
        else:
            print(f"Format de fichier non supporté : {file_extension}")
            return

        # Configurer la reconnaissance
```

```
        config = speech.RecognitionConfig(
            encoding=encoding,
            sample_rate_hertz=sample_rate,
            audio_channel_count=channels,
            language_code="en-US",  # À définir selon votre logique
        )

        with open(audio_file, "rb") as f:
            audio_content = f.read()

        audio = speech.RecognitionAudio(content=audio_content)

        # Effectuer la reconnaissance vocale de longue durée
        try:
            operation = client.long_running_recognize(config=config, audio=audio)
            response = operation.result(timeout=300)  # Ajuster le timeout si nécessaire
        except Exception as e:
            print(f"Erreur lors de la transcription : {e}")
            return

        print(response.results)

        transcription = ""
        for result in response.results:
            transcription += result.alternatives[0].transcript + "\n"

        with open(output_filename, "w", encoding="utf-8") as f:
            f.write(transcription)
        print(f"Transcription écrite dans {output_filename}")

    except Exception as e:
        print(f"Une erreur s'est produite lors de la génération de la transcription pour {output_filename} : +


def process_audio_files(input_dir, output_dir):
    os.makedirs(output_dir, exist_ok=True)

    all_audio_files = [f for f in os.listdir(input_dir) if f.endswith(('.mp3', '.wav', '.m4a'))]
    total_files = len(all_audio_files)
    print(f"Total des fichiers audio à traiter : {total_files}")
```

```
    if total_files == 0:
        print(f"Aucun fichier audio trouvé dans le répertoire '{input_dir}'.")
        return

    files_processed = 0



    for filename in all_audio_files:
        audio_file_path = os.path.join(input_dir, filename)
        output_filename = os.path.join(output_dir, f"{os.path.splitext(filename)[0]}.txt")
        if os.path.exists(output_filename):
            print(f"Ignorer {filename} : {output_filename} existe déjà.")
            continue
        print(f"\nTraitement {files_processed + 1}/{total_files} : {filename}")
        try:
            # Déterminer la langue en fonction du suffixe du fichier
            if filename.endswith('-zh.mp3') or filename.endswith('-zh.wav') or filename.endswith('-zh.m4a'):
                language_code = "cmn-CN"
            else:
                language_code = "en-US"

            # Mettre à jour la configuration dans speech_to_text si nécessaire
            # Pour simplifier, nous définirons le language_code dans config dans speech_to_text

            speech_to_text(
                audio_file=audio_file_path,
                output_filename=output_filename,
            )
            files_processed += 1
            print(f"Fichier {files_processed}/{total_files} traité.\n")
        except Exception as e:
            print(f"Échec du traitement de {filename} : {e}")
            continue

    print(f"Traitement terminé ! {files_processed}/{total_files} fichiers traités.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Traiter les fichiers audio pour générer des transcriptions."
    parser.add_argument('--input_dir', type=str, default="assets/audios", help="Répertoire d'entrée pour les f
```

```
args = parser.parse_args()



process_audio_files(
    input_dir=args.input_dir,
    output_dir=OUTPUT_DIRECTORY,
)
```