Génération Audio de Conversation

Prompt :

```
faites au moins 100 tours de conversation couvrant tous les détails de ce PDF, donnez-moi un format JSON à pro
```

```
[
    {
      "speaker": "A",
      "line": "Hey, j'entends beaucoup parler de Machine Learning (ML), Deep Learning (DL) et GPT ces dernier
    },
    {
      "speaker": "B",
      "line": "Bien sûr ! Commençons par les bases. Le Machine Learning est un domaine de l'informatique où le
    }
]
```

Code :

```python
import os
import json
import random
import subprocess
from google.cloud import texttospeech
import tempfile
import time
import argparse

# Répertoire de sortie fixe pour les conversations
OUTPUT_DIRECTORY = "assets/conversations"
INPUT_DIRECTORY = "scripts/conversation"

def text_to_speech(text, output_filename, voice_name=None):
    print(f"Génération de l'audio pour : {output_filename}")
    try:
        client = texttospeech.TextToSpeechClient()
        synthesis_input = texttospeech.SynthesisInput(text=text)
        if not voice_name:
            voice_name = random.choice(["en-US-Journey-D", "en-US-Journey-F", "en-US-Journey-O"])
        voice = texttospeech.VoiceSelectionParams(language_code="en-US", name=voice_name)
        audio_config = texttospeech.AudioConfig(
            audio_encoding=texttospeech.AudioEncoding.MP3,
            effects_profile_id=["small-bluetooth-speaker-class-device"]
        )

        retries = 5
        for attempt in range(1, retries + 1):
            try:
                response = client.synthesize_speech(input=synthesis_input, voice=voice, audio_config=audio_con
                with open(output_filename, 'wb') as out:
                    out.write(response.audio_content)
                print(f"Contenu audio écrit dans {output_filename}")
                return True
            except Exception as e:
                print(f"Erreur lors de la tentative {attempt} : {e}")
                if attempt == retries:
                    print(f"Échec de la génération de l'audio après {retries} tentatives.")
                    return False
                wait_time = 2 ** attempt
                print(f"Nouvelle tentative dans {wait_time} secondes...")
                time.sleep(wait_time)
    except Exception as e:
        print(f"Une erreur s'est produite lors de la génération de l'audio pour {output_filename} : {e}")
        return False

def process_conversation(filename):
    filepath = os.path.join(INPUT_DIRECTORY, filename)
    output_filename = os.path.join(OUTPUT_DIRECTORY, os.path.splitext(filename)[0] + ".mp3")

    if os.path.exists(output_filename):
        print(f"Le fichier audio existe déjà : {output_filename}")
        return

    try:
        with open(filepath, 'r', encoding='utf-8') as f:
            conversation = json.load(f)
    except Exception as e:
        print(f"Erreur lors du chargement du fichier de conversation {filename} : {e}")
        return
```

```
temp_files = []

voice_name_A = random.choice(["en-US-Wavenet-D", "en-US-Wavenet-E", "en-US-Wavenet-F"])
voice_name_B = random.choice(["en-US-Studio-O", "en-US-Studio-M", "en-US-Studio-Q"])

for idx, line_data in enumerate(conversation):
    speaker = line_data.get("speaker")
    line = line_data.get("line")
    if not line:
        continue
    temp_file = os.path.join(OUTPUT_DIRECTORY, f"temp_{idx}.mp3")
    temp_files.append(temp_file)

    voice_name = None
    if speaker == "A":
        voice_name = voice_name_A
    elif speaker == "B":
        voice_name = voice_name_B

    if not text_to_speech(line, temp_file, voice_name=voice_name):
        print(f"Échec de la génération de l'audio pour la ligne {idx+1} de {filename}")
        # Nettoyage des fichiers temporaires
        for temp_file_to_remove in temp_files:
            if os.path.exists(temp_file_to_remove):
                os.remove(temp_file_to_remove)
        return

if not temp_files:
    print(f"Aucun audio généré pour {filename}")
    return

# Concaténation avec ffmpeg
concat_file = os.path.join(OUTPUT_DIRECTORY, "concat.txt")
with open(concat_file, 'w') as f:
    for temp_file in temp_files:
        f.write(f"file '{os.path.abspath(temp_file)}'\n")

try:
    subprocess.run(
        ['ffmpeg', '-f', 'concat', '-safe', '0', '-i', concat_file, '-c', 'copy', output_filename],
            check=True,
            capture_output=True
        )
        print(f"Concaténation réussie de l'audio dans {output_filename}")
    except subprocess.CalledProcessError as e:
        print(f"Erreur lors de la concaténation de l'audio : {e.stderr.decode()}")
    finally:
        os.remove(concat_file)
        for temp_file in temp_files:
            os.remove(temp_file)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Traite les fichiers JSON de conversation pour générer de l'a
    args = parser.parse_args()

    os.makedirs(OUTPUT_DIRECTORY, exist_ok=True)

    for filename in os.listdir(INPUT_DIRECTORY):
        if filename.endswith(".json"):
            process_conversation(filename)
```