对话音频生成

提示:

至少进行100轮对话，涵盖关于此PDF的所有细节，并给我提供此PDF的JSON格式。

```
[
    {
      "speaker": "A",
      "line": "嘿，我最近听到很多关于机器学习（ML）、深度学习（DL）和GPT的内容。你能给我解释一下吗？"
    },
    {
      "speaker": "B",
      "line": "当然！我们从基础开始。机器学习是计算机科学的一个领域，系统通过数据学习以提高性能，而无需显式编程。可以把
    }
]
```

代码:

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

# 固定的输出目录用于对话
OUTPUT_DIRECTORY = "assets/conversations"
INPUT_DIRECTORY = "scripts/conversation"

def text_to_speech(text, output_filename, voice_name=None):
    print(f" 正在生成音频: {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"音频内容已写入 {output_filename}")
                return True
            except Exception as e:
                print(f"第 {attempt} 次尝试出错：{e}")
                if attempt == retries:
                    print(f"经过 {retries} 次尝试后，音频生成失败。")
                    return False
                wait_time = 2 ** attempt
                print(f"等待 {wait_time} 秒后重试...")
                time.sleep(wait_time)
    except Exception as e:
        print(f"生成 {output_filename} 的音频时发生错误：{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"音频文件已存在：{output_filename}")
        return

    try:
        with open(filepath, 'r', encoding='utf-8') as f:
            conversation = json.load(f)
    except Exception as e:
        print(f"加载对话文件 {filename} 时出错：{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" 生成 {filename} 的第 {idx+1} 行音频失败")
        # 清理临时文件
        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" 未生成 {filename} 的音频")
    return

# 使用 ffmpeg 进行拼接
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" 成功将音频拼接至 {output_filename}")
    except subprocess.CalledProcessError as e:
        print(f" 拼接音频时出错: {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=" 处理对话 JSON 文件以生成音频。")
    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)
```