Google Cloud Speech-to-Text

Recientemente experimenté con la API de Speech-to-Text de Google Cloud. A continuación, presento una función en Python que utilicé para realizar la transcripción.

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

# Directorio de salida fijo
OUTPUT_DIRECTORY = "assets/transcriptions"



def speech_to_text(audio_file, output_filename):
    print(f"Generando transcripción para: {output_filename}")
    try:
        client = speech.SpeechClient()

        # Cargar archivo de audio con pydub para determinar parámetros
        audio_segment = AudioSegment.from_file(audio_file)
        sample_rate = audio_segment.frame_rate
        channels = audio_segment.channels

        # Determinar la codificación basada en la extensión del archivo
        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"Formato de archivo no soportado: {file_extension}")
            return

        # Configurar el reconocimiento
```

```
        config = speech.RecognitionConfig(
            encoding=encoding,
            sample_rate_hertz=sample_rate,
            audio_channel_count=channels,
            language_code="en-US",  # Configurar según tu lógica
        )

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

        audio = speech.RecognitionAudio(content=audio_content)

        # Realizar reconocimiento de voz de larga duración
        try:
            operation = client.long_running_recognize(config=config, audio=audio)
            response = operation.result(timeout=300)  # Ajustar el tiempo de espera según sea necesario
        except Exception as e:
            print(f"Error durante la transcripción: {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"Transcripción escrita en {output_filename}")

    except Exception as e:
        print(f"Ocurrió un error al generar la transcripción para {output_filename}: {e}")


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 de archivos de audio a procesar: {total_files}")
```

```
    if total_files == 0:
        print(f"No se encontraron archivos de audio en el directorio '{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"Omitiendo {filename}: {output_filename} ya existe.")
            continue
        print(f"\nProcesando {files_processed + 1}/{total_files}: {filename}")
        try:
            # Determinar el idioma basado en el sufijo del nombre del archivo
            if filename.endswith('-zh.mp3') or filename.endswith('-zh.wav') or filename.endswith('-zh.m4a'):
                language_code = "cmn-CN"
            else:
                language_code = "en-US"

            # Actualizar la configuración en speech_to_text si es necesario
            # Por simplicidad, estableceremos el language_code en config dentro de speech_to_text

            speech_to_text(
                audio_file=audio_file_path,
                output_filename=output_filename,
            )
            files_processed += 1
            print(f"Archivo {files_processed}/{total_files} procesado.\n")
        except Exception as e:
            print(f"Error al procesar {filename}: {e}")
            continue

    print(f"¡Procesamiento completado! {files_processed}/{total_files} archivos procesados.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Procesar archivos de audio para generar transcripciones.")
    parser.add_argument('--input_dir', type=str, default="assets/audios", help="Directorio de entrada para los
```

```
args = parser.parse_args()



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