

# Google Cloud Speech-to-Text

I recently experimented with Google Cloud's Speech-to-Text API. The following is a Python function I used to perform the transcription.

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

# Fixed output directory
OUTPUT_DIRECTORY = "assets/transcriptions"

def speech_to_text(audio_file, output_filename):
    print(f"Generating transcription for: {output_filename}")
    try:
        client = speech.SpeechClient()

        # Load audio file with pydub to determine parameters
        audio_segment = AudioSegment.from_file(audio_file)
        sample_rate = audio_segment.frame_rate
        channels = audio_segment.channels

        # Determine encoding based on file extension
        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"Unsupported file format: {file_extension}")
            return

        # Configure recognition
```

```

config = speech.RecognitionConfig(
    encoding=encoding,
    sample_rate_hertz=sample_rate,
    audio_channel_count=channels,
    language_code="en-US", # Set based on your logic
)

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

audio = speech.RecognitionAudio(content=audio_content)

# Perform long-running speech recognition
try:
    operation = client.long_running_recognize(config=config, audio=audio)
    response = operation.result(timeout=300) # Adjust timeout as needed
except Exception as e:
    print(f"Error during 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 written to {output_filename}")

except Exception as e:
    print(f"An error occurred while generating transcription for {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 audio files to process: {total_files}")

```

```

if total_files == 0:
    print(f"No audio files found in '{input_dir}' directory.")
    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"Skipping {filename}: {output_filename} already exists.")
        continue
    print(f"\nProcessing {files_processed + 1}/{total_files}: {filename}")
    try:
        # Determine language based on filename suffix
        if filename.endswith('-zh.mp3') or filename.endswith('-zh.wav') or filename.endswith('-zh.m4a'):
            language_code = "cmn-CN"
        else:
            language_code = "en-US"

        # Update the config in speech_to_text if needed
        # For simplicity, we'll set the language_code in config within speech_to_text

        speech_to_text(
            audio_file=audio_file_path,
            output_filename=output_filename,
        )
        files_processed += 1
        print(f"File {files_processed}/{total_files} processed.\n")
    except Exception as e:
        print(f"Failed to process {filename}: {e}")
        continue

print(f"Processing complete! {files_processed}/{total_files} files processed.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Process audio files to generate transcriptions.")
    parser.add_argument('--input_dir', type=str, default="assets/audios", help="Input directory for audio file")

```

```
args = parser.parse_args()

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