SPEECH TO TEXT

(python)

Overview

This documentation provides an overview of an audio processing system, including its architecture, system dependencies, and algorithms. The system is designed to perform speech recognition on different types of audio inputs: real-time microphone input, pre-recorded audio files, and long-duration audio files.

Architecture

The system consists of three main components:

Real-time Audio Recognition (app.py):

- Captures audio from a microphone in real-time.
- Performs noise adjustment to improve recognition accuracy.
- Transcribes the recorded audio to text using Google's Speech Recognition API.

Pre-recorded Audio File Recognition (app_audio.py):

- Processes pre-recorded audio files.
- Uses the Speech Recognition API to convert audio content into text.

Long-duration Audio File Recognition (long_audio.py):

- Handles long audio files by splitting them into smaller chunks based on silence detection.
- Each chunk is processed individually to transcribe the audio content into text.
- Uses the pydub library for audio manipulation and chunking.

System Dependencies

- The system relies on several key dependencies, primarily Python libraries, to function correctly:

speech_recognition: For interfacing with the Google Speech Recognition API to convert audio to text.
pydub: For audio file manipulation, including loading, exporting, and splitting audio files based on silence.
os: For file path manipulations (used in long audio processing).

To install the required dependencies, you can use the following pip commands:

```
pip install speechrecognition
pip install pydub
```

Algorithms

1. Real-time Audio Recognition (app.py)

- Noise Adjustment:

Uses recognizer.adjust_for_ambient_noise(source, duration=1) to adjust for ambient noise.

- Recording Audio:

Captures 4 seconds of audio using recognizer.listen(source, timeout=4).

- Speech Recognition:

Converts recorded audio to text using recognizer.recognize_google(recorded_audio, language="en-US").

2. Pre-recorded Audio File Recognition (app_audio.py)

- Recording Audio:

Loads the audio file using sr.AudioFile("/path/to/audio.wav") and listens to it using recognizer.listen(source).

- Speech Recognition:

Converts the audio content to text using recognizer.recognize_google(recorded_audio, language="en-US").

3. Long-duration Audio File Recognition (long_audio.py)

- Loading and Splitting Audio:

Loads the long audio file using AudioSegment.from_mp3(filename).

Splits the audio into chunks based on silence using split_on_silence(long_audio, min_silence_len=1800, silence_thresh=-17).

- <u>Processing Chunks:</u>

Iterates through each audio chunk, exports it to a temporary file in WAV format, and processes it using the Speech Recognition API.
Handles recognition exceptions to ensure the process continues despite errors in individual chunks.

Real-time Audio Recognition

```
import speech_recognition as sr

recognizer = sr.Recognizer()

with sr.Microphone() as source:
    print("Adjusting noise")
    recognizer.adjust_for_ambient_noise(source, duration=1)
    print("Recording for 4 seconds")
    recorded_audio = recognizer.listen(source, timeout=4)
    print("Done recording")

try:
    print("Recognizing the text")
    text = recognizer.recognize_google(recorded_audio, language="en-US")
    print("Decoded Text: {}".format(text))
except Exception as ex:
    print(ex)
```

Pre-recorded Audio File Recognition:

```
import speech_recognition as sr

recognizer = sr.Recognizer()

with sr.AudioFile("/path/to/audio.wav") as source:
    recorded_audio = recognizer.listen(source)
    print("Done recording")

try:
    print("Recognizing the text")
    text = recognizer.recognize_google(recorded_audio, language="en-US")
    print("Decoded Text: {}".format(text))
```

```
except Exception as ex:
    print(ex)
```

Long-duration Audio File Recognition:

```
import os
from pydub import AudioSegment
import speech_recognition as sr
from pydub.silence import split_on_silence

recognizer = sr.Recognizer()

def load_chunks(filename):
    long_audio = AudioSegment.from_mp3(filename)
    audio_chunks = split_on_silence(long_audio, min_silence_len=1800, silence_thresh=-17)
    return audio_chunks

for audio_chunk in load_chunks('./sample_audio/long_audio.mp3'):
    audio_chunk.export("temp", format="wav")
    with sr.AudioFile("temp") as source:
        audio = recognizer.listen(source)
        try:
            text = recognizer.recognize_google(audio)
            print("Chunk: {}".format(text))
        except Exception as ex:
            print("Error occurred")
            print(ex)

print("Processing complete")
```

Conclusion

This system provides a comprehensive solution for speech recognition from various audio sources. By leveraging the speech_recognition and pydub libraries, it can handle real-time audio, pre-recorded audio files, and long-duration audio files effectively.