Project Report

My project topic is real-time speech recognition implementation. There are four main functions of my implementation. First is to recognize speech from microphone. Second is real-time recognize speech from microphone. The third function is to recognize speech from an input wave file. Fourth is in real-time recognize speech from an input wave file.

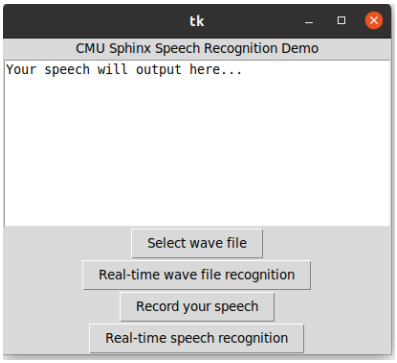
There are a lot of open-source speech recognition algorithms. For this project, I chose the CMU Sphinx[2]. It is an open-source speech recognition project written in C, which is developed at Carnegie Mellon University. The accuracy for Sphinx speech recognition is high compared to traditional methods like Dynamic Time Wrapping. However, compared to new methods including deep learning using models like Bert-Net, the accuracy is not as high. The advantage of CMU Sphinx is that is consumes less resources and CPU power. It does not need a high-level platform and in fact it can be used in Embedded Systems. Nevertheless, if the speaker speaks a good American accent, the accuracy is still very high. The library I use is speech_recognition[1],

which is a wrapper for Sphinx model.

The project is implemented in Python3, and the library I am using is speech_recognition[1] (The installation guide can be reached here). I implemented an API called SpeechRecognition, which contains three methods for microphone input, file input, and binary object input separately. You just simply call getStringFromMicrophone, getStringFromFile, getStringFromBin, and they will return a string object for the speech. The first two methods are not in real-time. However, using getStringFromBin, I can implement it in real-time.

```
import speech_recognition as sr

class SpeechRecognition(object):
    r = sr.Recognizer()

    def __SphinxHandler(self, audio):
        try:
            print('Sphinx thinks you said: "' + self.r.recognize_sphinx(audio) + '"')
            return self.r.recognize_sphinx(audio)
        except sr.UnknownValueError:
            print("Sphinx could not understand audio")
        except sr.RequestError as e:
            print("Sphinx error; {0}".format(e))

    def getStringFromMicrophone(self):
        with sr.Microphone() as source:
            self.r.adjust_for_ambient_noise(source)
            print("Please speak...")
            audio = self.r.listen(source)
            print("End of speech detected...")
            return self.__SphinxHandler(audio)

    def getStringFromFile(self, file):
        with sr.AudioFile(file) as source:
            audio = self.r.record(source)
            return self.__SphinxHandler(audio)

    def getStringFromBin(self, data, rate, width):
        audio = sr.AudioData(data, rate, width)
        return self.__SphinxHandler(audio)
```

For demo part, I implemented a simple UI using tkinter. It is a window with a text field and four buttons. The speech will be presented in text-field in both real-time and not in real-time.



In the demo implementation, I mainly implemented four functions using the methods provided by SpeechRecognition. I will mainly talk about the implementation for the two real-time functions.

```
def realTimeWaveFile():
    BLOCKLEN = 30000
    default_dir = "/Users/QY_Tang/Dropbox/NYU/DSP Lab/project/english.wav"
    filePath = Tk.filedialog.askopenfilename(title='File Selection', initialdir=(path.expanduser(default_dir)))
    if filePath == '':
        return
    _thread.start_new_thread(helper.playWaveFile, (filePath, ))

    wf = wave.open( filePath, 'rb')
    LEN = wf.getnframes()
    audioData = np.asarray(np.int16([]))
    num_blocks = int(math.floor(LEN  / BLOCKLEN))
    res = ""
    for i in range(0, num_blocks):
        input_bytes = wf.readframes(BLOCKLEN)
        input_tuple = struct.unpack('h' * BLOCKLEN, input_bytes)
        audioSeg = np.asarray(np.int16(input_tuple))
        audioData = np.concatenate((audioData, audioSeg), axis=0)
        sr.getStringFromBin(audioData, wf.getframerate(), wf.getsampwidth())
    res = sr.getStringFromBin(audioData, wf.getframerate(), wf.getsampwidth())
    text.delete(1.0, Tk.END)
    text.insert(1.0, res)
    wf.close()
```

This is for playing the real-time wave file function. Where I open up a file and read BLOCKLEN frames at one time. Then the input_bytes needs to be transformed into tuple, and tuple is transformed into np.array to fit into getStringFromBin.

```
37 def realTimeMicrophone():
38     BLOCKLEN = 15000
39     WIDTH       = 2         # Number of bytes per sample
40     CHANNELS    = 1         # mono
41     RATE        = 16000     # Sampling rate (frames/second)
42     DURATION    = 6         # duration of processing (seconds)
43     N = int(DURATION * RATE / BLOCKLEN)     # N : Number of samples to process
44 
45     p = pyaudio.PyAudio()
46 
47     stream = p.open(
48         format      = p.get_format_from_width(WIDTH),
49         channels    = CHANNELS,
50         rate        = RATE,
51         input       = True,
52         output      = False)
53     audioData = np.asarray(np.int16([]))
54     print("Please speak now!")
55     for n in range(0, N):
56         input_bytes = stream.read(BLOCKLEN, exception_on_overflow = False)
57         input_tuple = struct.unpack('h' * BLOCKLEN, input_bytes)
58         audioSeg = np.asarray(np.int16(input_tuple))
59         audioData = np.concatenate((audioData, audioSeg), axis=0)
60         _thread.start_new_thread(sr.getStringFromBin, (audioData, RATE, WIDTH,))
61     res = sr.getStringFromBin(audioData, RATE, WIDTH)
62     text.delete(1.0, Tk.END)
63     text.insert(1.0, res)
64     p.terminate()
```

This is the implementation of real-time speech recognition from microphone. It is similar to the wave file. It just get frames from microphone and then it is the same.

```
66 def fileSelect():
67     default_dir = "/Users/QY_Tang/Dropbox/NYU/DSP Lab/project/author.wav"
68     filePath = Tk.filedialog.askopenfilename(title='File Selection', initialdir=(path.expanduser(default_dir)))
69     if filePath == '':
70         return
71     _thread.start_new_thread(helper.playWaveFile, (filePath, ))
72     res = sr.getStringFromFile(filePath)
73     text.delete(1.0, Tk.END)
74     text.insert(1.0, res)
75 
76 
77 def microphoneRecognize():
78     res = sr.getStringFromMicrophone()
79     text.delete(1.0, Tk.END)
80     text.insert(1.0, res)
```

These two functions is for the non-real-time implementation. It turns out that the outcome for the demo is pretty good, and please see the demo video for further reference.

References

[1] https://github.com/Uberi/speech_recognition

[2] https://cmusphinx.github.io