HCI Assignment One

1750114 程子鸣

HCI Assignment One
- Product Overview
  - Product Name
  - Environments
  - Dependencies
  - Basic Functionalities
  - How To Use
  - Usage Notes
- GUI Changes
  - Corresponding Code Changes
- Main Functional Codes
- Highlights
- How to Improve the Recognition Precision

Product Overview

Product Name

> Naive "大白" Voice Assistant
>
> (●—●)

Environments

- Developed using MBP with Baidu Python SDK
- Need stable network connection to work
- "Open Apps" only works on MacOS systmes and may cast errors on other platforms

Dependencies

PyQt5 Pyaudio wave aip pydub

Basic Functionalities

1. Talking to the Master and repeat what the Master had said

```
<<"滴滴"
>>重复一下我说了什么
<<你说了“重复一下我说了什么”
```

2. Open Siri

```
>>打开Siri
>>open siri
<<你说了“....”
<<open /Applications/Siri.app
```

3. Master can order "大白" to open music players

   In this product, the default music player is NeteaseMusic

```
>>听音乐
>>打开网易云音乐
>>网易云音乐
<<你说了"..."
<<open /Applications/NeteaseMusic.app
```

4. Change modes

   Master can change "大白" between Chinese and English mode.

```
>>切换为英文模式
<<你说了"..."
<<[English Mode]
>>change to chinese mode
<<you said "..."
<<[中文模式]
```

5. Waking up

   After openning other apps, "大白" will fall asleep. Yet you can wake it up by saying sentences contains the word "大白"

```
>>...大白...
<<[大白苏醒]
```

6. Automatically Shut down

   If the Master accidentally opened "大白", after three empty voice imput, "大白" will be closed automatically.

```
>>""
>>""
>>""
<<[大白退出]
```

7. Quit "大白"

```
>>退出
>>拜拜
>>再见
>>关闭大白
<<[大白退出]
```

How To Use

```
python3 asr.py
```

Usage Notes

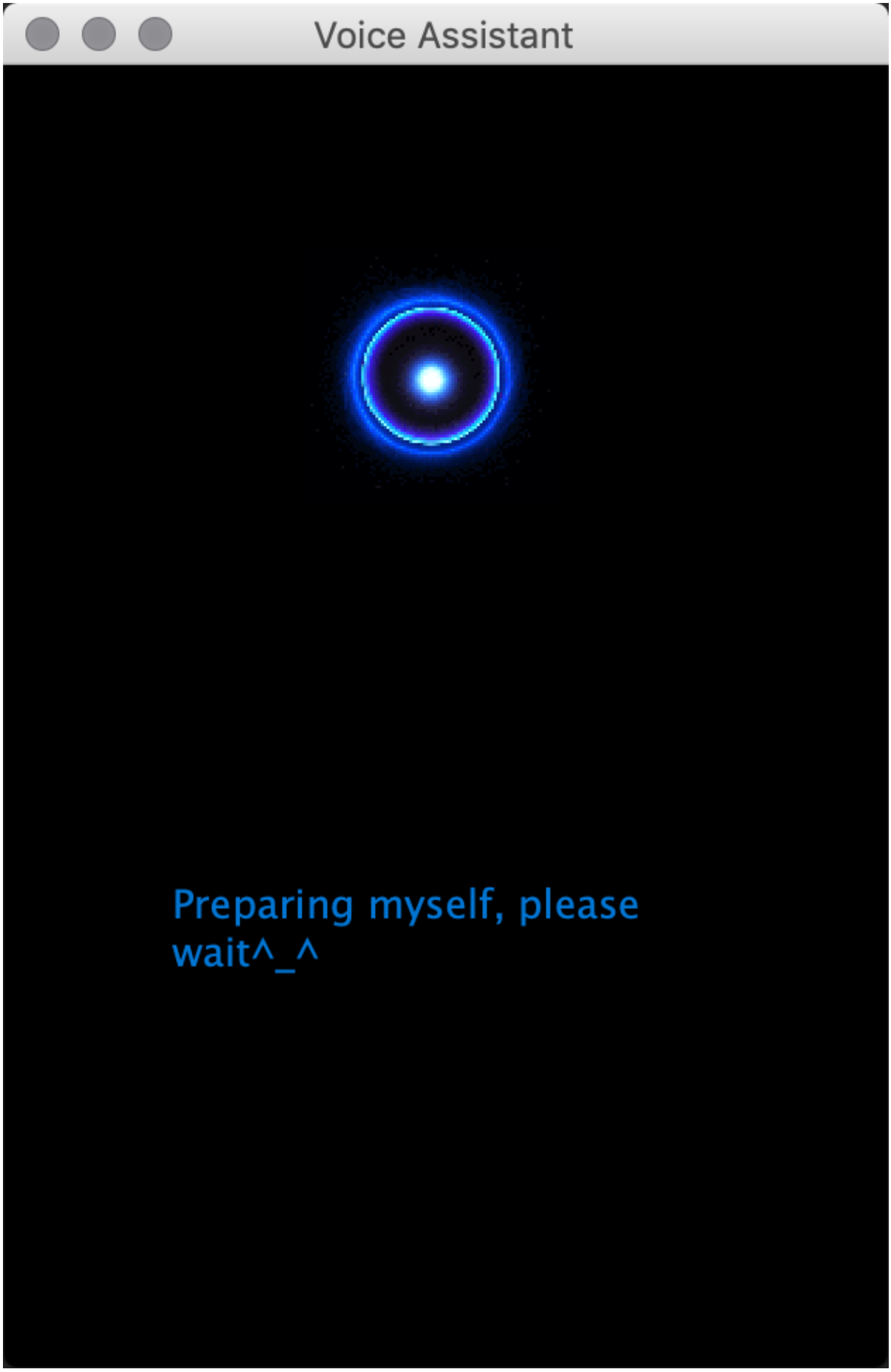
1. Master should speak after each prompt "滴滴".
2. Though "大白" is tolerant of ambient noise to some extent, extremely noisy environments which obscures Masters' voice input are not supported.
3. Your voice will be recorded and saved as temporary wavs into the "database/" folder in order to be uploaded to Baidu Yuyin Services. Feel free to clear that folder.
4. Do not delete or move "temp/" folder. "大白" will initiate itself and download voice files every time it . These voice files is stored in the "temp/" folder.

GUI Changes

1. After "大白" making the prompt sound, the background color of main window will change to dark purple.
2. In Chinese mode, the title of the main window is "大白" and all labels will be translated into Chinese , while in English mode the title is "Dabai" and all labels will be in English.

> Recording

3. Once "大白" is launched, there will be a short pause for "大白" to be prepared.



4. Thanks to "Qthread", "大白" will keep listening to you in the background so that there is no need for any extra buttons to wake up "大白" or to start your conversations. Just speak shortly after "大白" says "滴滴" and wake it up by calling its name ^_^

Corresponding Code Changes

```
# 1
# asrinterface.py
def changeColor(self, MainWindow, rgb_tuple):
    rgb_tuple = map(lambda x: str(x), rgb_tuple)
    tuple_string = "("+",".join(rgb_tuple)+")"
    MainWindow.setStyleSheet("background-color: rgb" + tuple_string + ";")

# voice_thread.py
# Class VoiceThread
color_signal = pyqtSignal(tuple)
# Class MainRecognizingThread
self.voiceThread.color_signal.connect(self.window.change_color)
self.color_signal.emit((64, 0, 60))
self.color_signal.emit((0, 0, 0))
```

```
# 2
# asrinterface.py
def retranslateUi(self, MainWindow, mode):
    if mode == "zh":
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "大白"))
        self.label_3.setText(_translate("MainWindow", "1. 说\"听音乐\"来享受音
乐"))
        self.label_2.setText(_translate("MainWindow", "试试:"))
        self.label.setText(_translate("MainWindow", "你好，有什么能帮你的?"))
        self.label_4.setText(_translate("MainWindow", "2. 需要更多帮助? \"打开
Siri\""))
    else:
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Dabai"))
        self.label_3.setText(_translate("MainWindow", "1. Enjoy music by saying
\"Play music\""))
        self.label_2.setText(_translate("MainWindow", "You can:"))
        self.label.setText(_translate("MainWindow", "Hi! How can I help?"))
        self.label_4.setText(_translate("MainWindow", "2. Get some help by
saying \"open siri\""))

# voice_thread.py
# Class VoiceThread
mode_signal = pyqtSignal()
# Class MainRecognizingThread
self.mode_signal.emit()
```

```
# 3
```

```
# asrinterface.py
def myretranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Voice Assistant"))
    self.label_3.setText(_translate("MainWindow", "Preparing myself, please
wait^_^"))

# First
self.myretranslateUi(MainWindow)
self.greetingThread.prepare_signal.connect(self.prepared_callback)self.prepare_
signal.emit(True)
# Next
# voice_thread.py
# Class MainRecognizingThread
prepare_signal = pyqtSignal(bool)
self.prepare_signal.emit(True)
self.prepare_signal.emit(False)
```

Main Functional Codes

> I intentionally use split my code into an additional file called "voice_thread.py" for convenience.

```
from PyQt5.QtCore import QThread, pyqtSignal
import speech_recognition
import pyaudio
from utils import *
import os

MY_NAME = "大白"


class VoiceThread(QThread):
    quit_signal = pyqtSignal()
    color_signal = pyqtSignal(tuple)
    mode_signal = pyqtSignal()

    def __init__(self):
        super().__init__()
        self.recognizer = speech_recognition.Recognizer()
        self.no_audio_times = 0
        self.mode = "zh"

    def run(self) -> None:
        while True:
            self.basic_loop()
            # 等待唤醒
```

```
            while True:
                code = self.record_once(True)
                with open(code, 'rb') as fp:
                    data = fp.read()
                    result = client.asr(data, 'wav', 16000, {
                        'dev_pid': 1536,
                    })
                    if result["err_no"] == 0 and result["err_msg"] ==
"success.":
                        line = " ".join(result["result"])
                        if line.find(MY_NAME) != -1:
                            play_sound("temp/wake_up.wav")
                            break

    def basic_loop(self):
        condition = True
        while condition:
            code = self.record_once(False)
            self.color_signal.emit((0, 0, 0))
            condition = self.recognize_once(code)
            self.sleep(1)


    def record_once(self, silence):
        if not silence:
            play_sound("temp/begin_listening.wav")
            self.color_signal.emit((64, 0, 60))
        pa = pyaudio.PyAudio()
        stream = pa.open(format=pyaudio.paInt16, channels=1,
                         rate=framerate, input=True,
                         frames_per_buffer=NUM_SAMPLES)
        my_buf = []
        count = 0
        log("Speech Input")
        while count < TIME * 15:  # 控制录音时间
            string_audio_data = stream.read(NUM_SAMPLES)
            my_buf.append(string_audio_data)
            count += 1
        code = activation_code(1, 16)
        filename = str(code[0]) + ".wav"
        save_wav_from_buffer(filename, my_buf)
        stream.close()
        return "database/" + filename


    def process_once(self, command):
        if command.find("英文模式") != -1:
            play_sound("temp/chinese_to_english.wav")
            self.mode = "en"
```

```
            self.mode_signal.emit()
            return True
        elif command.lower().find("chinese mode") != -1:
            play_sound("temp/english_to_chinese.wav")
            self.mode = "zh"
            self.mode_signal.emit()
            return True
        elif command.lower().find("siri") != -1:
            play_sound("temp/open_siri.wav")
            os.system("open /Applications/Siri.app")
            return False
        elif command.lower().find("网易云音乐") != -1 or command.lower().find("音
乐") != -1 or \
                command.lower().find("music") != -1:
            play_sound("temp/play_music.wav")
            os.system("open /Applications/NeteaseMusic.app")
            return False
        elif (command.lower().find("再") != -1 and command.lower().find("见")
!= -1) or \
                command.lower().find("退出") != -1 or \
                (command.lower().find("关闭") != -1 and command.lower().find("大
白") != -1) \
                or command.lower().find("拜拜") != -1:
            play_sound("temp/quit.wav")
            self.quit_signal.emit()
        else:
            return True

    def recognize_once(self, filePath):
        with open(filePath, 'rb') as fp:
            data = fp.read()


        # 识别本地文件
        if self.mode == "zh":
            result = client.asr(data, 'wav', 16000, {
                'dev_pid': 1536,
            })
        else:
            result = client.asr(data, 'wav', 16000, {
                'dev_pid': 1737,
            })
        if result["err_no"] == 0 and result["err_msg"] == "success.":
            line = " ".join(result["result"])
            if self.mode == "zh":

                result = client.synthesis("你说了" + line, self.mode, 1, {
                    'vol': 5,
                    'per': 1
```

```
                })
            else:
                result = client.synthesis("you said" + line, "zh", 1, {
                    'vol': 5,
                    'per': 1
                })
            prepared = save_synthesis("temp", result)
            if prepared:
                play_sound("temp/temp.wav")
            else:
                play_sound("temp/recoginition_failed.wav")
            self.no_audio_times = 0
            success = self.process_once(line)
            return success
        elif result["err_no"] == 3300 or result["err_no"] == 3301 or
result["err_no"] == 3302:
            play_sound("temp/no_sound.wav")
            self.no_audio_times += 1
            if self.no_audio_times > 2:
                self.sleep(1)
                play_sound("temp/no_sound_annoy.wav")
                self.quit_signal.emit()
                self.terminate()
            else:
                return True
        else:
            self.no_audio_times = 0
            play_sound("temp/recoginition_failed.wav")
            return True


class MainRecognizingThread(QThread):
    prepare_signal = pyqtSignal(bool)

    def __init__(self, window):
        super().__init__()
        self.voiceThread = None
        self.prepared = None
        self.window = window

    def run(self) -> None:
        self.sleep(3)
        # welcoming
        self.prepare()
        # background recognizing
        self.voiceThread = VoiceThread()
        self.voiceThread.quit_signal.connect(self.window.close_window)
        self.voiceThread.color_signal.connect(self.window.change_color)
        self.voiceThread.mode_signal.connect(self.window.change_mode)
```

```
        self.voiceThread.start()
        self.voiceThread.wait()

    def greeting(self) -> None:
        pass

    def prepare(self):
        if not os.path.exists("temp"):
            os.mkdir("temp")
        self.prepared = True
        result = client.synthesis('你好呀呀呀呀呀,冲冲冲', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        # 识别正确返回语音二进制 错误则返回dict 参照下面错误码
        self.prepared = save_synthesis("hello", result)
        result = client.synthesis('滴滴', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("begin_listening", result)
        result = client.synthesis('咋办，我没听懂你说了啥，你要不再说一次', 'zh', 1,
{
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("recoginition_failed", result)
        result = client.synthesis('你咋不说话呢，麦坏了', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("no_sound", result)
        result = client.synthesis('哼,一直不说话,拜拜了您馁', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("no_sound_annoy", result)
        result = client.synthesis('巴啦啦魔法变身，英文模式', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("chinese_to_english", result)
        result = client.synthesis('papa mamamiya sakura ohno，中文模式', 'zh',
1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("english_to_chinese", result)
        result = client.synthesis('好的，立马唤醒大哥', 'zh', 1, {
```

```
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("open_siri", result)
        result = client.synthesis('好的，你先听着歌，我歇会', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("play_music", result)
        result = client.synthesis('诶，你喊我呢，啥事儿啊', 'zh', 1, {
            'vol': 5,
            'per': 1
        })
        self.prepared = save_synthesis("wake_up", result)
        result = client.synthesis('拜拜', 'zh', 1, {
            'vol': 7,
            'per': 1,
            'spd': 3
        })
        self.prepared = save_synthesis("quit", result)
        result = client.synthesis('你好呀呀呀呀呀,冲冲冲,和我聊天吧', 'zh', 1, {
            'vol': 5,
            'spd': 6,
            'per': 1
        })
        self.prepared = save_synthesis("baigei", result)
        if self.prepared:
            playSoundThread = PlaySoundThread("temp/baigei.wav")
            playSoundThread.start()
            self.sleep(3)
            self.prepare_signal.emit(True)
            playSoundThread.wait()
        else:
            self.prepare_signal.emit(False)


class PlaySoundThread(QThread):
    def __init__(self, filename):
        super().__init__()
        self.filename = filename

    def run(self) -> None:
        play_sound(self.filename)
```

> Moreover, I also used some utility functions, such as save wav files from bytes and play audios from wav files. I put those functions in "utils.py"

```
import time
import pyaudio
```

```
import wave
from aip import AipSpeech
import random
import string
from pydub.audio_segment import AudioSegment
import os


# Settings
APP_ID = "16070198"
API_KEY = "rNFMfKG2duelK45ZuZgbTTXU"
SECRET_KEY = "gLXVE7fpuYHPaBtPxl1yRLt3e5p4KC8T"

RATE = "16000"
FORMAT = "wav"

framerate=16000
NUM_SAMPLES=2000
channels=1
sampwidth=2
TIME=2

# BaiduYuYin Client
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)


def log(text):
    print("MySiri app log:", time.asctime(time.localtime(time.time())), " log:
", text)


def play_sound(filename):
    CHUNK = 1024
    wf = wave.open(filename, 'rb')

    p = pyaudio.PyAudio()

    stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                    channels=wf.getnchannels(),
                    rate=wf.getframerate(),
                    output=True)

    data = wf.readframes(CHUNK)

    while data != b'':
        stream.write(data)
        data = wf.readframes(CHUNK)

    stream.stop_stream()
```

```
    stream.close()

    p.terminate()


def save_wav_from_buffer(filename, buffer):
    if not os.path.exists("database"):
        os.mkdir("database")
    base_dir = "database/"
    framerate = 16000
    f = wave.open(base_dir + filename, "wb")
    f.setnchannels(1)
    f.setsampwidth(2)
    f.setframerate(framerate)
    # 将wav_data转换为二进制数据写入文件
    f.writeframes(b"".join(buffer))
    f.close()


def save_synthesis(filename, result):
    if not isinstance(result, dict):
        base_path = "temp/"
        with open(base_path + filename + '.mp3', 'wb') as f:
            f.write(result)
        sound = AudioSegment.from_mp3(base_path + filename + ".mp3")
        sound.export(base_path + filename + ".wav", format='wav')
        return True
    else:
        return False


# 激活码生成
def activation_code(count, length):
    base = string.ascii_uppercase + string.ascii_lowercase + string.digits    #
生成激活码可能包含的字符集（大写字母、小写字母、数字）
    return [''.join(random.sample(base, length)) for i in range(count)]
```

Highlights

1. "大白" uses voice synthesis to provide a more natural conversational experience just like talking to a real person. Voice synthesis also brings a better experience through necessary prompt audios than texts printed in the console or screen.
2. "大白" supports both Chinese and English.
3. "大白" will automatically pause when you opened another apps through "大白". Later, "大白" could be woken up unless "大白" is manually closed by the Master.
4. "大白" will automatically exit after three blank audio input if your audio environment are not

desirable or "大白" launched unexpectedly.

5. Master can quit "大白" without clicking a single button, just say "拜拜", "退出", "关闭大白", "再见"
6. All audios will be saved under the database folder so that developers can check them for further information (as a database for speech-related research).

How to Improve the Recognition Precision

> Technically speaking, we can simply substitute other excellent speech recognition SDKs or APIs for the original Speech_Recognition module. That's what I have done for this project. I use Baidu's Python SDK for higher recognition precision.
>
> Meanwhile, we can apply some noise reduction or user-voice highlighting algorithms to enhance the contrast between noises and user voices.
>
> On some special occasions, we may need to customize our recognition models. Therefore, we should train our own customized recognition models by using HMM or other trainning methods.
>
> Moreover, we can also use several different APIs or SDKs provided by different companies and then vote for the same audio input to generate a more robust result.