HCI Lab 1 Report

ID	Name
1953902	GAO Yangfan

1. Basic Topics

a. Running the original code

Different from what Prof. Shen mentioned in the instruction slide, I chose a simpler way, which only uses `pip` and `conda` instead of installing from source, to install the package needed for running the demo. Here's what I did:

1. Installing SpeechRecognition:

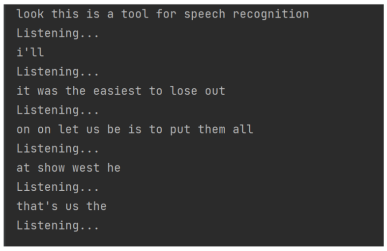
Just executing `pip install speechrecognition` is OK for this.

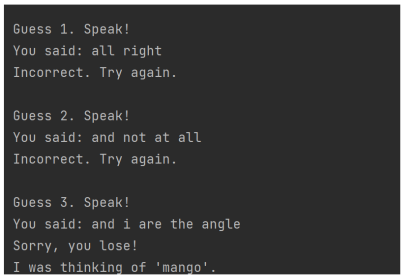
2. Installing PyAudio & PocketSphinx

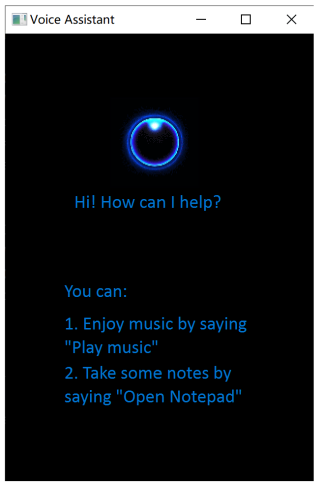
To install PyAudio, simply run `pip install pyaudio` is OK, this is also a pre-requisite for PocketSphinx.

After this, I tried to run `pip install pocketsphinx` to install PocketSphinx directly but failed. I had to install `swig` by `conda install swig` and then I successfully installed PocketSphinx by `pip install pocketsphinx`.

Then I ran `test.py` `guessTheWord.py` and `asr.py`, here are the results







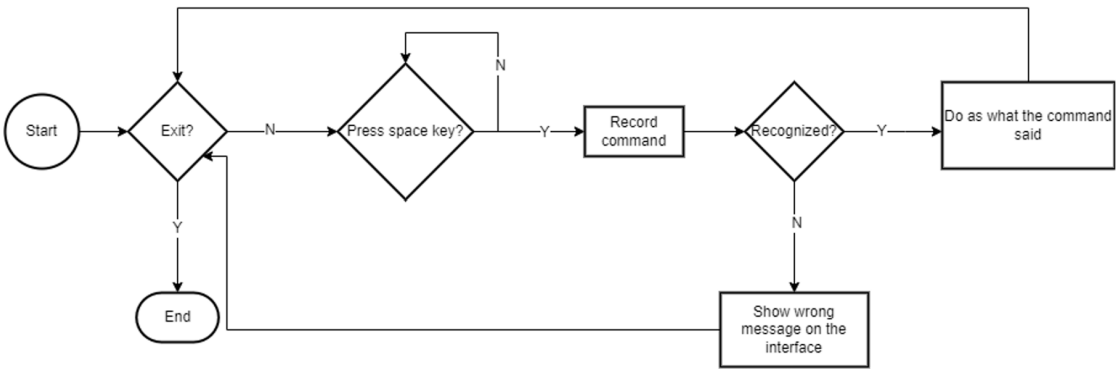
I have to say that the accuracy is unbelievably low and I will discuss approaches that may improve this in advanced topics.

b. Implementing the required functions

I modified `asr.py` and `asrinterface.py` to implement the required functions. Here are what I did concretely:

1. Modified the code in `asr.py` to implement the logic for ASR and execute what the user commands.
2. Modified the code in `asrinterface.py` to add some more tips for the interface
3. Applied multi-thread technique to make sure that the interface and the process of ASR work independently
4. Applied signal technique to dynamically change the content in widgets.

Here is the flow chart of this ASR application

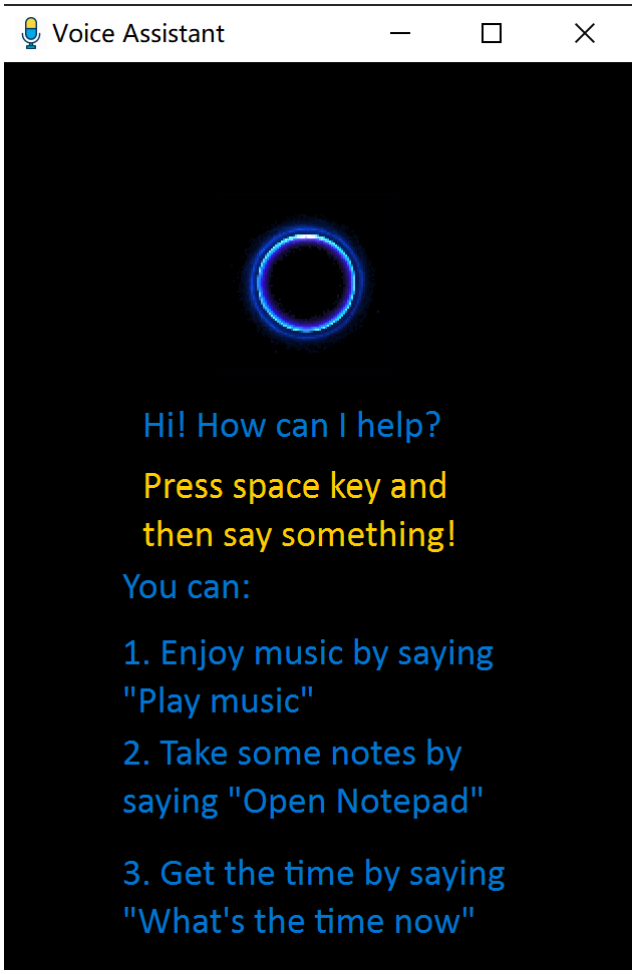


The modified interface is shown below.

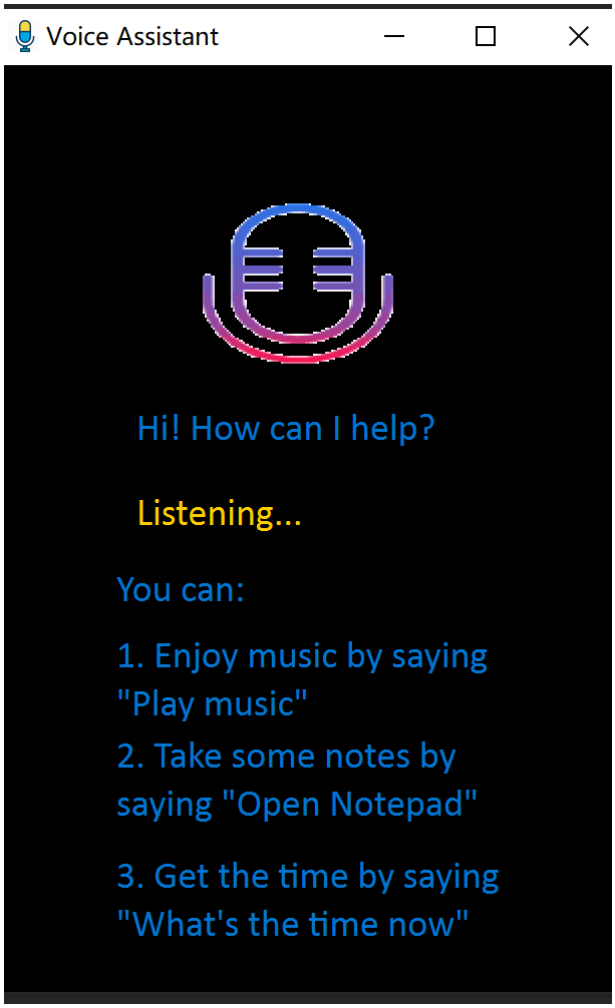
It can be seen that besides the required functions(picture 3 and 4), I **added an interaction `QLabel` widget** (picture 1) colored orange at the center of this interface and **added a function that shows the current time** (picture 5) when users say "What's the time now" or other commands like this. Also, I made full use of the images and GIFs that Prof. Shen provided by **adding an icon for the app** and **changed the GIF displayed at the top when recording the**

commands (picture 2). Moreover, I **handled the situation when the command can't be understood and give friendly reminders to users** (picture 6)

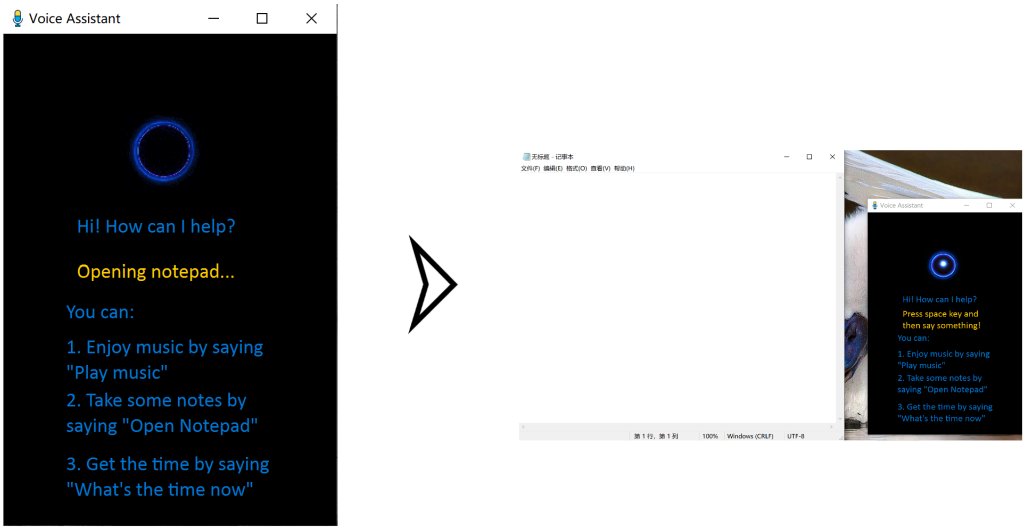
1. The initial interface



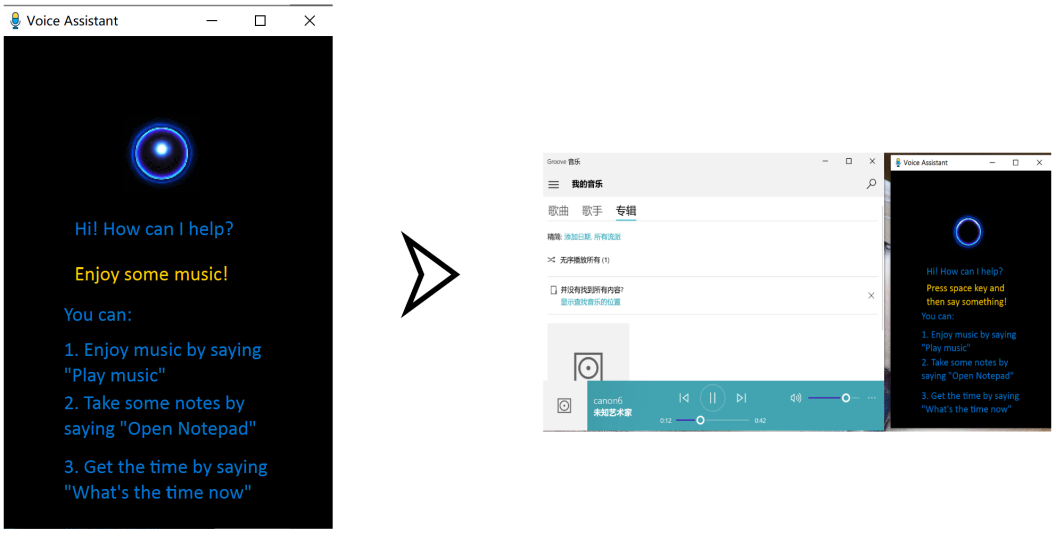
2. When listening to the user



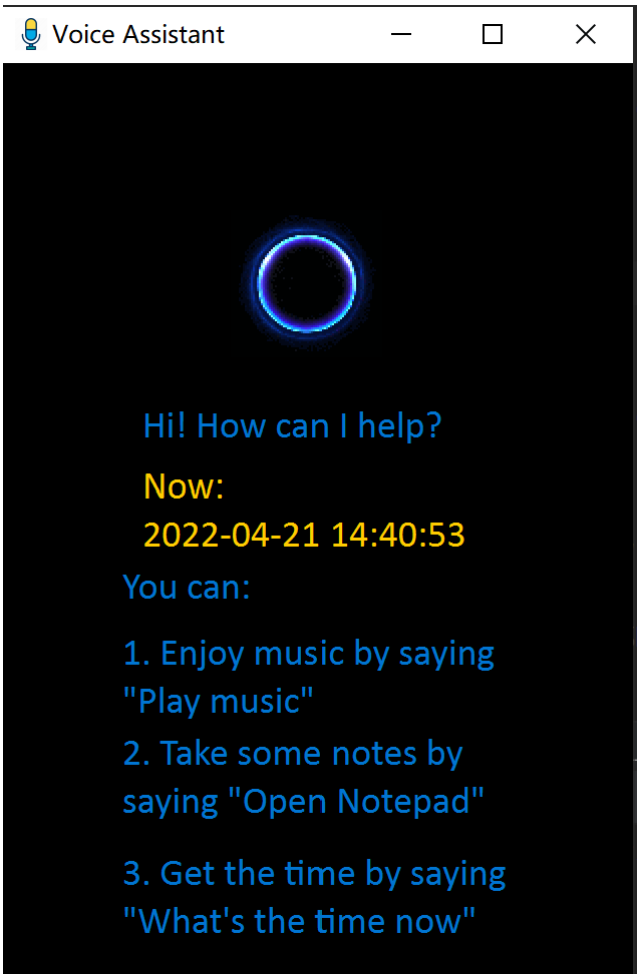
3. When opening the notepad



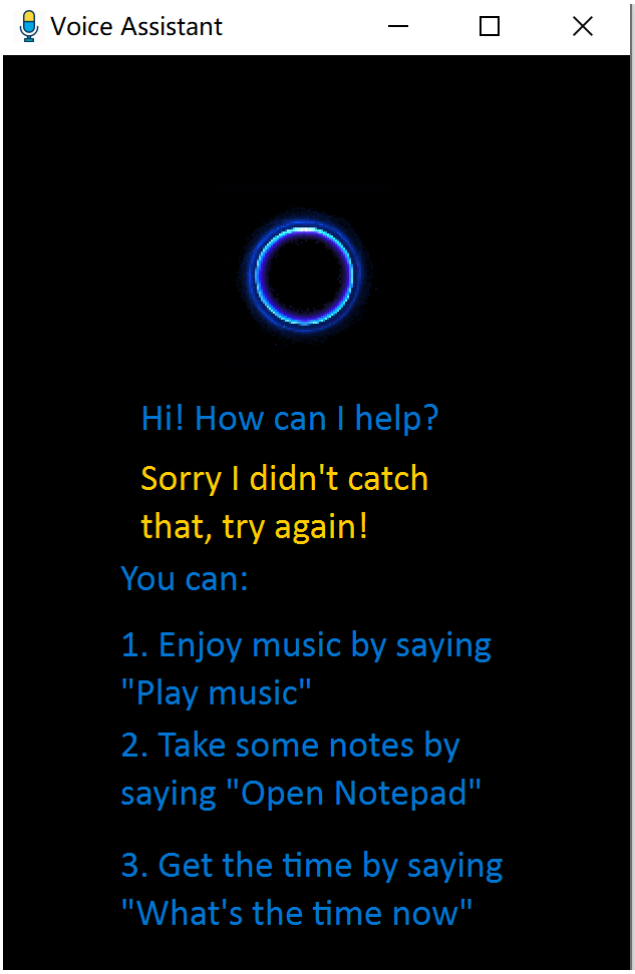
4. When playing music



5. When displaying the time now



6. Didn't understand what the user said:



2. Advanced Topics

a. Improving the accuracy of ASR & action of the agent

As I used the offline ASR API PocketSphinx to implement the functions, the accuracy of ASR in this application is very low. So I applied some tricks to improve the accuracy without using other APIs. Here are what I did.

1. **fine-tuning**

Before doing this homework, I did some research on SpeechRecognition library and discovered there are 2 functions that can be fine-tuned. One is `adjust_for_ambient_noise` that reduces the influence of ambient noise and the other is `listen` that records the command. Concretely, I tuned the parameters like this:

```
self.recognizer.adjust_for_ambient_noise(source, duration=0.5)
audio = self.recognizer.listen(source, timeout=10, phrase_time_limit=3)
```

the tuned parameters are:

1. `duration` in `adjust_for_ambient_noise`: default 1, decreasing this will improve the effect of noise-reducing.
2. `timeout` in `listen`: tuning this parameter will limit the maximum time for the program to run without any audio inputs, reducing the probability of recording noise
3. `phrase_time_limit` in `listen`: the maximum time of recording a command, decreasing this will also reduce the possibility of recording noise.

2. **syllable recognition instead of word recognition**

After some experiments on PocketSphinx, I discovered that although the accuracy of recognizing a single word is very low, it can catch some syllables of the words correctly. As the commands of the implemented functions are independent in terms of pronunciation, I applied syllable recognition to this to improve the accuracy, concretely, I wrote code like:

```
                if sentence.__contains__('pad') or sentence.__contains__('note')
or sentence.__contains__('pa'):
                    global_signal.text_change.emit('Opening notepad...')
                    time.sleep(1)
                    win32api.ShellExecute(0, 'open', 'notepad.exe', '', '', 1)
```

Besides recognizing 'note' and 'pad', I added recognizing the syllable 'pa' to improve the accuracy of action of the agent

b. Multi-lingual support (optional)

PocketSphinx also supports Chinese ASR, I gave it a try. Here's what I did

1. download the Chinese language packge here
2. unzip the package and put the directory in the pocketsphinx-data directory of speechrecognition package
3. wrote `test_zh-CN.py` to test Chinese ASR

Here is the result:

```
D:\miniconda\envs\lane-recognition\python.exe D:/COURSE_WORK_Bachelor/HCI2022Spring/labs/lab1/lab1-asr/test_zh-cn.py
正在倾听
当然 开局 时 的

进程已结束，退出代码为 0
```

Actually I said '打开记事本'

The accuracy is still low in terms of Chinese ASR. As Chinese,which is based on characters, is unlike English, which is based on syllables, I cannot use the tricks above to improve the action accuracy because there are too many homophonic characters. So the only approach is to change APIs like Baidu ASR API.

Then a Chinese ASR agent can be implemented like above, I didn't do this repeated work.

3.How to run this

1. run `config.sh`
2. run `python asr.py`
3. Press space and then speak something!
4. (optional) If you want to try Chinese ASR, follow steps 1,2 in section 2.b and run `test_zh-CN.py`