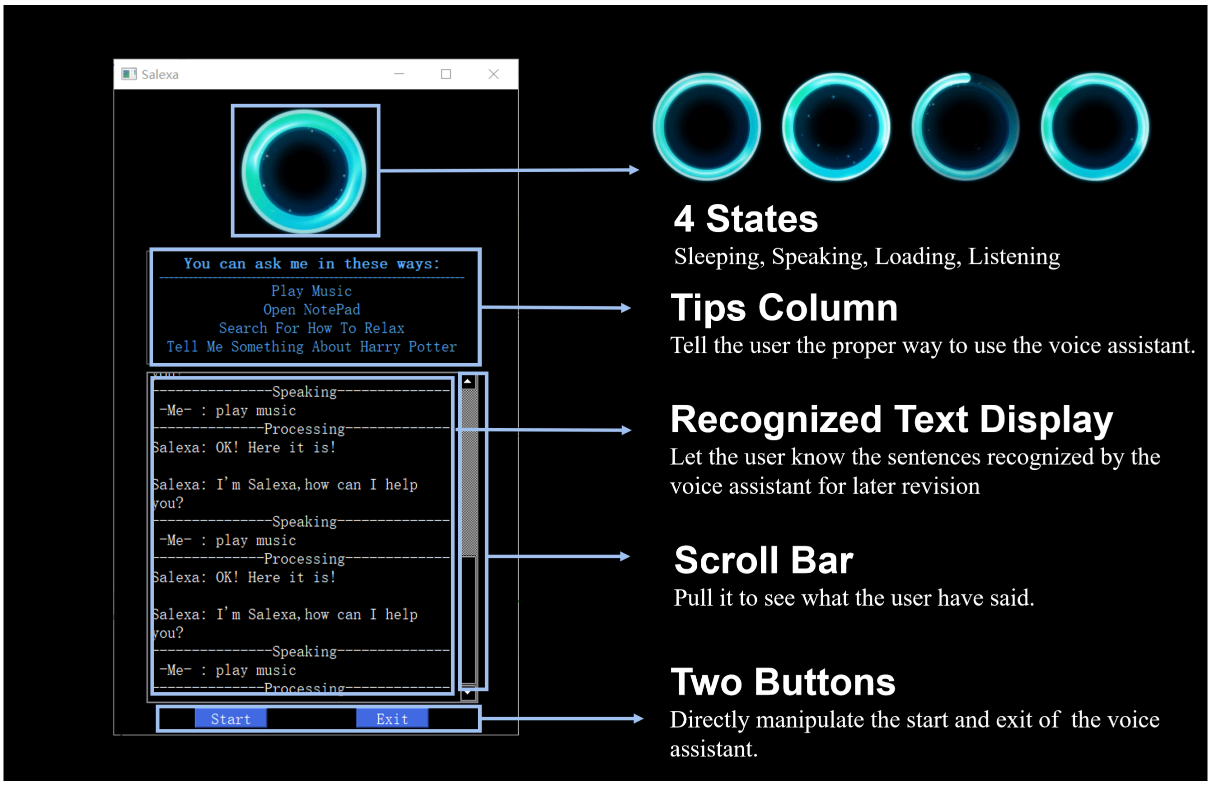
Lab 1: Automatic Speech Recognition

Name	ID
Weida Wang	2151300

1. Modification

1.1. GUI


Salexa
4 States
Sleeping, Speaking, Loading, Listening
Tips Column
Tell the user the proper way to use the voice assistant.
Recognized Text Display
Let the user know the sentences recognized by the voice assistant for later revision
Scroll Bar
Pull it to see what the user have said.
Two Buttons
Directly manipulate the start and exit of the voice assistant.
Start
Exit

```
You can ask me in these ways:
Play Music
Open NotePad
Search For How To Relax
Tell Me Something About Harry Potter
```

```
---------------Speaking---------------
-Me- : play music
---------------Processing---------------
Salexa: OK! Here it is!

Salexa: I'm Salexa,how can I help
you?
---------------Speaking---------------
-Me- : play music
---------------Processing---------------
Salexa: OK! Here it is!

Salexa: I'm Salexa,how can I help
you?
---------------Speaking---------------
-Me- : play music
---------------Processing---------------
```

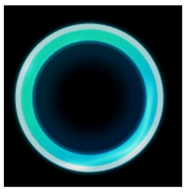
1.1.1. Two Button

Using two buttons to directly manipulate the start and end of the conversation.

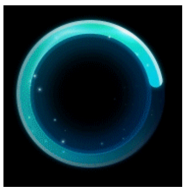
Start - To start the voice assistant.
Exit - To close the voice assistant.

1.1.2. Four States

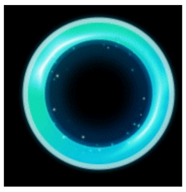
Using four different gifs to distinguish between the four different states of the voice assistant.



Sleeping
- Waiting for the user to wake it up by clicking the start button.
- Waiting for break time to end.



Loading
- The voice assistant is recognizing commands and performing related operations.



Speaking
- The voice assistant is reading out the execution results.

Listening
- Listening what the user says.

1.1.3. Others

Scroll Bar

Pull it to see what the user have said.

Tips Column

Tell the user the proper way to use the voice assistant.

Two Beeps

Tell the user about the recorder begin and end statement.

Recognized Text Display

Let the user know the sentences recognized by the voice assistant for later revision

1.2. Code

1.2.1. Class

- `class Salex` : Implementing the basic functions of voice assistant.
- `class MyThread(QThread)` : Manipulate the entire program by using a multi-threaded approach.
- `class UI_Form(object)` : Implement UI interface design.
- `class MyWindw(QWidget,UI_Form)` : Achieve overall control of the page.

1.2.2. UI Page Design

A new set of UI interfaces were designed with the aid of Qt Designer, while using open source gifs for the different states.

```
class UI_Form(object):

    def process(self):
        self.thread = MyThread()
        self.thread.text_signal.connect(self.textEdit.append)
        self.thread.gif_signal.connect(self.change_gif)
        self.thread.start()

    def change_gif(self, str):
        self.gif.stop()
        self.gif = QMovie(str)
        self.label.setMovie(self.gif)
        self.gif.start()

    def setup_ui(self, Form):
        if not Form.objectName():
            Form.setObjectName(u"Form")
        Form.resize(500, 800)
        Form.setStyleSheet("background-color: rgb(0, 0, 0);")

        # gif label(changeable)
        self.label = QLabel(Form)
        self.label.setObjectName(u"label")
        self.label.setGeometry(QRect(150, 20, 171, 161))
        self.gif = QMovie("icon/normal.gif")
        self.label.setMovie(self.gif)
        self.gif.start()

        # Prompt box
        self.textBrowser = QTextBrowser(Form)
        self.textBrowser.setObjectName(u"textBrowser")
        self.textBrowser.setGeometry(QRect(40, 200, 411, 141))

        # Conversation box
        self.scrollArea = QScrollArea(Form)
        self.scrollArea.setObjectName(u"scrollArea")
        self.scrollArea.setGeometry(QRect(40, 350, 411, 411))
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setStyleSheet("background-color: black; color: white;font-size: 20px;")
        self.scrollArea.verticalScrollBar().setStyleSheet(
            "QScrollBar:vertical {background-color: black; color: white;}")
        # Text box
        self.textEdit = QTextEdit(self.scrollArea)
        self.textEdit.setGeometry(QRect(0, 0, 409, 409))
        self.textEdit.setPlainText("")

        self.scrollArea.setWidget(self.textEdit)

        # Start button
        self.pushButton_Start = QPushButton("<b>Start</b>", Form)
        self.pushButton_Start.setObjectName(u"pushButton")
        self.pushButton_Start.setGeometry(QRect(100, 767, 90, 25))
        self.pushButton_Start.setStyleSheet("background-color: #4169E1; color: white;font-size: 20px;")
        self.pushButton_Start.clicked.connect(self.process)

        # Exit button
        self.pushButton_Exit = QPushButton("<b>Exit</b>", Form)
        self.pushButton_Exit.setObjectName(u"pushButton")
        self.pushButton_Exit.setGeometry(QRect(300, 767, 90, 25))
        self.pushButton_Exit.setStyleSheet("background-color: #4169E1; color: white;font-size: 20px;")
        self.pushButton_Exit.clicked.connect(QApplication.quit)

        self.retranslate_ui(Form)
        QMetaObject.connectSlotsByName(Form)
        # setup_ui

    # generated by Qt Designer
    def retranslate_ui(self, Form):
        Form.setWindowTitle(QCoreApplication.translate("Form", u"Salexa", None))
        self.label.setText("")
        self.textBrowser.setHtml(QCoreApplication.translate("Form",
          u"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
          "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
          "p, li { white-space: pre-wrap; }\n"
          "</style></head><body style=\" font-family:'SimSun'; font-size:9pt; font-weight:400; font-style:normal;\">\n"
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
          "12pt; color:#55aaff;\">Play Music</span></p>\n"
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
          "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text
           None))
        self.pushButton_Exit.setText(QCoreApplication.translate("Form", u"Exit", None))
        self.pushButton_Start.setText(QCoreApplication.translate("Form", u"Start", None))
        # retranslateUi
```

1.2.3. Changing Gifs

Salexa forms different gifs in different states, while using a special signal and slot mechanism in PyQt5 to update the UI interface in real time.

```
# defined in class MyThread
def run(self):
        s_words = "Salexa: "
        u_words = " -Me- : "
        while True:
            # Send a signal to replace the GIF every event
            # Salexa reads the text to start recognizition
            text = s_words + "I'm Salexa,how can I help you?"
            self.gif_signal.emit("icon/play.gif")
            self.text_signal.emit(text)
            self.salexa.start()
            # User speaking
            self.text_signal.emit("---------------Speaking--------------")
            self.gif_signal.emit("icon/listen.gif")
            cmd = self.salexa.get_audio()
            self.text_signal.emit(u_words + cmd)
            # Prcessing the command
            self.text_signal.emit("--------------Processing-------------")
            self.gif_signal.emit("icon/loading.gif")
            sleep(3)
            text = self.salexa.run(cmd)
            self.text_signal.emit(s_words + text)

            # Acess if the conversation is over
            if text == "Fine, see you next time!":
                self.gif_signal.emit("icon/normal.gif")
                break

            # Take a short break
            self.text_signal.emit("")
            self.gif_signal.emit("icon/normal.gif")
            sleep(3)
```

1.2.4. Voice Recognition

A range of library functions are called to implement different functions, listed as follows.

- Combining `gtts.gTTS` with `playsound.playsound` to let Salexa read out the text.
- Using `speech_recognition.Recognizer.listen` to record the audio from the user, and using `speech_recognition.Recognizer.recognize_google` to change the audio into text.
- Using `win32api.ShellExecute` to open NotePad and play music.
- `pywhatkit.search` are used to search on Google.

```
class Salexa:
    # Salexa read out the text by using gtts.TTS
    def speak(self, text):
        language = "en"
        output = gTTS(text=text, lang=language, slow=False)
        output.save("./sounds/output.mp3")
        playsound("./sounds/output.mp3")
        return text

    # Record what the user said with two beeps to remind the user
    # Using Google Recognizer for better experience.
    def get_audio(self):
        recorder = sr.Recognizer()
        with Microphone() as source:
            playsound("./sounds/alert.mp3")  # alert sound start speaking
            recorder.adjust_for_ambient_noise(source)
            audio = recorder.listen(source)
            playsound("./sounds/alert.mp3")  # alert sound end speaking
            print("Processing...")
            text = recorder.recognize_google(audio)
            print(f"You said:{text}")
        return text

    def run(self, text):
        if "play" in text.lower() and "music" in text.lower():
            rtext = self.case_music()
        elif "notepad" in text.lower():
            rtext = self.case_notepad()
        elif "stop" in text.lower() or "exit" in text.lower():
            rtext = self.case_exit()
        else:
            rtext = self.case_search(text)
        return rtext

    def start(self):
        text = "I'm Salexa,how can I help you?"
        self.speak(text)
        print(text)
        return text

    # play music
    def case_music(self):
        text = "OK! Here it is!"
        self.speak(text)
        ShellExecute(0, 'open', 'music.mp3', '', '', 1)
        return text
    # open notepad
    def case_notepad(self):
        text = "OK! Just opened!"
        self.speak(text)
        ShellExecute(0, 'open', 'notepad.exe', '', '', 1)
        return text
    # search on the internet
    def case_search(self, text):
        stext = "Here's what I found"
        self.speak(stext)
        search(text.replace("search for", ""))
        return stext
    # end the conversation
    def case_exit(self):
        text = "Fine, see you next time!"
        self.speak(text)
        return text
```

2. Accuracy

2.1. Experiment

2.1.1. Sphinx

Run with the code `text = recorder.recognize_sphinx(audio)` .

The test was conducted in a quiet and noisy environment respectively, by pronouncing the words by either myself (American English with CHinese accent) or by using the pronunciation method of Youdao Dictionary(authentic American pronunciation)，and get the experimental results as follows. (The test vocabulary includes common instructions, such as “Hello”, ”Open Notepad”,”Play music”,”Search for Harry Potter” etc. 20 times each.)

Accuracy	Noisy	Queit
Myself	0.1	0.20
Youdao Dictionary	0.25	0.35

2.1.2. Google

Run with the code `text = recorder.recognize_google(audio)` .

After doing the same experiments as Sphinx, I get the results below.

Accuracy	Noisy	Queit
Myself	0.80	0.95
Youdao Dictionary	0.90	0.95

2.2. Improvement

2.2.1. Conclusions

By analyzing the above experimental findings, it is easy to obtain the following conclusions.

- The quieter the environment, the clearer the user's pronunciation and the more accurate the voice assistant's recognition.
- Online Google speech recognition significantly outperforms offline Sphinx.

2.2.2. Potential Improvement

Based on the experiments and conclusions above, I propose several possible ways to improve the accuracy of speech recognition listed as follows.

- **Speak in a quiet place.** This reduces background noise and improves the accuracy of speech recognition.You can speak in a soundproof room or use items such as curtains or carpets to absorb excess sound.
- **Speak as clearly and precisely as possible.** This makes it easier for speech recognition technology to catch your words, so speak clearly and loudly. You should also pay attention to your pronunciation and intonation to avoid slurring or speaking too fast.
- **Change a better recogniser.** This will improve the quality and efficiency of your speech recognition, as different software (or package) has different features and benefits. You can choose a speech recognition software (or package) that suits your language, accent and scene, or use a speech recognition tool that can learn and adapt itself. In my experiment, I just changed the Sphinx to Google for a better experience.