2.6 Voice Interaction

1. What is Voice Interaction?

Voice Interaction (VI) refers to a method of communication between humans and computers or devices through spoken language. It integrates speech recognition and speech synthesis, enabling devices to both understand user commands and respond naturally, creating true two-way voice communication. To achieve natural voice interaction, factors such as semantic understanding and sentiment analysis must also be considered, allowing the system to accurately interpret user intent and provide appropriate responses.

This approach can be used as the foundation for developing our own AI assistant features.

2. How It Works

First, the wake word detection module listens for a specific wake-up word. Once detected, it initiates audio recording. After recording, Automatic Speech Recognition (ASR) converts the audio into text, which is then sent to a Large Language Model (LLM) to generate an appropriate response. The generated text is subsequently converted into speech through a Text-to-Speech (TTS) module and played back to the user. This entire process enables seamless and natural interaction between the user and the voice assistant.

3. Experiment Steps

1) Power on the device and connect to it via MobaXterm (refer to Appendix "5.1 Remote Connection Tools and Instructions" for connection guidance).

2) To check the microphone's port number, first disconnect the microphone and run the command. Then reconnect the microphone and run the command again to determine the port number (Note: do not connect any other USB devices during this process).

ll /dev | grep USB

```
> ll /dev | grep USB
```

- After disconnecting the microphone, no USB device should appear.

```
> ll /dev | grep USB
```

- Upon reconnecting the microphone, a USB port (e.g., ttyCH341USB1) will be listed (make sure to note this device name). The device name may vary depending on the main controller.

3) Navigate to the program directory:

cd large_models/

```
cd large_models/
```

4) Open the configuration file to enter your API Key. After editing, press Esc, then type :wq and hit Enter to save and exit:

vim config.py

```
9   llm_api_key = ''
10  llm_base_url = 'https://api.openai.com/v1'
11  os.environ["OPENAI_API_KEY"] = llm_api_key
```

5) Enter the port number you obtained and modify the corresponding microphone port settings for either WonderEcho Pro or the six-microphone setup. Uncomment the configuration for the port you intend to use and comment out the settings for any unused ports.

```
vim openai_interaciton_demo.py
```

If you are using the WonderEcho Pro, modify the corresponding section:

If you are using the 6-Microphone Array, modify the relevant section:

```
15  port = '/dev/ttyCH341USB1'
16  # kws = awake.WonderEchoPro(port)
17  kws = awake.CircleMic(port)
```

6) Run the program:

```
python3 openai_interaciton_demo.py
```

```
python3 openai_interaciton_demo.py
```

7) To stop the program at any time, simply press Ctrl+C.

4. Function Realization

After successful execution, the voice device will announce 'I'm ready.' Then, upon hearing the wake-up word 'HELLO_HIWONDER,' the device will respond with 'I'm here,' indicating that the assistant has been successfully awakened. You can now ask the AI assistant any questions:

For example: 'What are some fun places to visit in New York?'

```
asr_result: What are some fun places to visit in New York?

llm response: New York City offers a diverse array of fun places to explore. Here are some highlights:

1. **Central Park**: An iconic green space perfect for picnics, leisurely walks, or boat rides. You can also visit the Central Park Zoo.

2. **Times Square**: Famous for its bright lights, theaters, and bustling energy, it's a must-see, especially at night.

3. **The Statue of Liberty and Ellis Island**: Take a ferry to visit these historic landmarks and learn about America's immigration history.

4. **The Metropolitan Museum of Art**: One of the largest and most prestigious art museums in the world, with a vast collection spanning thousands of years.

5. **Brooklyn Bridge**: Walk or bike across this historic bridge for stunning views of the Manhattan skyline.

6. **The High Line**: A unique elevated park built on a former railway track, offering gardens, art installations, and great views of the city.

7. **Empire State Building**: Visit the observation deck for panoramic views of New York City.

8. **Chelsea Market**: A food hall and shopping mall that's perfect for foodies, with a variety of gourmet options.

9. **The Museum of Modern Art (MoMA)**: A world-renowned museum showcasing contemporary and modern art.

10. **Broadway**: Catch a show in the Theater District for an unforgettable evening of live performances.

11. **SoHo and Greenwich Village**: Explore trendy shops, art galleries, and vibrant nightlife in these iconic neighborhoods.

12. **Coney Island**: Enjoy amusement rides, classic boardwalk games, and delicious hot dogs in the summer.

13. **9/11 Memorial & Museum**: A poignant memorial and museum honoring the victims of the September 11 attacks.

14. **Staten Island Ferry**: Take a free ride for great views of the Statue of Liberty and the Manhattan skyline.

15. **Museum of Natural History**: Discover fascinating exhibits, including dinosaur fossils and the iconic planetarium.

These sites provide a mix of cultural, historical, and purely fun experiences that reflect the vibrant character of New York City!
Time to first byte: 1.94 seconds
```

5. Brief Program Analysis

The program integrates voice recognition, speech synthesis, and intelligent response functionalities to create a voice assistant. Interaction is initiated through the wake-up word (HELLO_HIWONDER). Users can converse with the assistant via voice commands, and the assistant will respond using text-to-speech technology. The overall structure is clear, with distinct modules that are easy to expand and maintain.

The source code for this program is located at:
/home/ubuntu/large_models/openai_interaction_demo.py

5.1 Module Import

```
import os
import time
from config import *
from speech import awake
from speech import speech
```

time: Used to control the interval between program executions.

speech: The core module, integrating wake-up word detection, speech

activity detection, speech recognition, TTS, and LLM.

5.2 Definition of Audio File Paths

```
wakeup_audio_path = './resources/audio/en/wakeup.wav'
start_audio_path = './resources/audio/en/start_audio.wav'
no_voice_audio_path = './resources/audio/en/no_voice.wav'
```

This section configures the audio file paths used by various functional modules, such as wake-up sounds, recording storage paths, and prompt sounds.

The text-to-speech (TTS) module is initialized to convert LLM responses into speech.

5.3 Main Functional Logic

```
def main():
    kws.start()
    while True:
        try:
            if kws.wakeup(): # 检测到唤醒词
                speech.play_audio(wakeup_audio_path)  # 唤醒播放
                asr_result = asr.asr() # 开启录音识别
                print('asr_result:', asr_result)
                if asr_result:
                    # 将识别结果传给智能体让他来回答
                    response = client.llm(asr_result, model='gpt-4o-mini')
                    print('llm response:', response)
                    tts.tts(response)
                else:
                    speech.play_audio(no_voice_audio_path)
            time.sleep(0.02)
        except KeyboardInterrupt:
            kws.exit()
            try:
                os.system('pinctrl FAN_PWM a0')
            except:
                pass
            break
        except BaseException as e:
            print(e)
```

Wake-up Detection: Continuously monitors for the wake-up word. Once detected, it stops the wake-up detection and plays the wake-up prompt sound.

Voice Processing: Records and recognizes the user's speech, uses the language model to generate a response, and then converts the response into speech for playback.

Error Handling: Catches exit signals and runtime errors to ensure the program exits safely and releases resources.