AI large model offline voice assistant

AI large model offline voice assistant
1. Online Voice Configuration
2. Creating a startup service (Systemd)
2.1 Create a startup script
2.2Creating a Systemd Service File
2.3Management and debugging services

Note: Since Jetson Nano is in Docker, the following method is only a guide and cannot be implemented directly

1. Online Voice Configuration

Before setting up auto-start, we must ensure that the program itself can work independently in an online state. This can be done by modifying the configuration file.

1. **Locate the configuration file**:
   In your project code, find and open the configuration file: `config/yahboom.yaml`
2. **Modify Configuration Parameters**:
   Please check the following parameters in the file and ensure that their values match those shown below. If the parameters do not exist, add them.

```
asr: # Voice node parameters
ros__parameters:
VAD_MODE: 2 # VAD sensitivity
sample_rate: 16000 # ASR audio sampling rate
frame_duration_ms: 30 # VAD frame size in milliseconds
use_oline_asr: True # Whether to use online ASR recognition (True for
online, False for offline)
mic_serial_port: "/dev/ttyUSB0" # Microphone serial port alias
mic_index: 0 # Microphone index
language: 'en' # ASR language

model_service: # Model server node parameters
ros__parameters:
language: 'en' # Large model interface language
useolinetts: True # Whether to use online speech synthesis (True for online,
False for offline)

# Large model configuration
#llm_platform: 'ollama' # Available platforms: 'ollama', 'tongyi', 'spark',
'qianfan', 'openrouter'
llm_platform: 'tongyi' # This example uses Tongyi as an example
```

   - `use_oline_asr` and `useolinetts` must be set to `tongyi`.
   - `llm_platform` must be set to `tongyi`.

     If you set it up like this, everything will be online.
3. **Save the file** and **recompile** the project to apply the changes:

```
cd ~/yahboom_ws
colcon build
source install/setup.bash
```

After completing this step, the program is already a purely online voice service.

2. Creating a startup service (Systemd)

Now, we will create a `systemd` service so that `largemodel_control.launch.py` will automatically run at system startup.

2.1 Create a startup script

In order for `systemd` to correctly load the ROS2 environment, the best practice is to create a simple `bash` script to encapsulate our startup command.

1. **Create script file**::
   In the directory (`~/yahboom_ws/src/largemodel/`), create a file named `start_largemodel.sh`.

   ```
   vim ~/yahboom_ws/src/largemodel/start_largemodel.sh
   ```

2. **Write script content**:
   Copy and paste the following content into the script file.

   ```
   #!/bin/bash

   # Source ROS2 Humble environment
   source /opt/ros/humble/setup.bash

   # Source Yahboom workspace environment
   source /home/jetson/yahboom_ws/install/setup.bash

   # Start the largemodel control script
   ros2 launch largemodel largemodel_control.launch.py
   ```

   Important: Please make sure to replace `/home/jetson/` in the script with your own user home directory path.

3. **Save and Exit**
4. **Give the script execution permissions**:

   ```
   chmod +x ~/yahboom_ws/src/largemodel/start_largemodel.sh
   ```

2.2Creating a Systemd Service File

This is the most important step. We will tell the system that we have a new service to manage.

1. **Create service file**:
   You will need `sudo` privileges to create this file.

   ```
   sudo vim /etc/systemd/system/largemodel.service
   ```

2. **Write service configuration**:
   Copy and paste the following content into the service file.

```
[Unit]
Description=Robot Service
After=network.target sound.target graphical.target multi-user.target
Wants=network.target sound.target graphical.target multi-user.target

[Service]
Type=simple
User=sunrise
Group=sunrise
Environment=DISPLAY=:0
Environment=XDG_RUNTIME_DIR=/run/user/1000
Environment=PULSE_SERVER=unix:/run/user/1000/pulse/native
SupplementaryGroups=audio video
ExecStartPre=/bin/sleep 10
ExecStart=/home/jetson/yahboom_ws/src/largemodel/start_largemodel.sh
Restart=on-failure
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

   !!! Extremely important!!!

   - Please make sure that the paths in `WorkingDirectory` and `ExecStart` are exactly the same as your actual paths.

3. **Save and exit**.

2.3Management and debugging services

Now that your service is created, we need to have systemd load it and set it to start automatically on boot.

1. **Reload the `systemd` daemon** so that it reads our newly created service file:

```
sudo systemctl daemon-reload
```

2. **Set the service to start automatically at boot**:

```
sudo systemctl enable largemodel.service
```

3. **Start the service immediately**:

```
sudo systemctl start largemodel.service
```

4. **Check service status**:
   This is the most important command to verify that the service is running successfully.

```
sudo systemctl status largemodel.service
``` * If you see `Active: active (running)`, congratulations! The service has started successfully!
* If the status is `failed` or another error, proceed to the next step for debugging.
```

5. **View service log (required for debugging)**:
   If the service fails to start, you can use the following command to view all real-time logs generated by the `ros2 launch` command, which is crucial for locating the problem.

   ```
   journalctl -u largemodel.service -f
   ```

After completing all the above steps, the purely online `largemodel` voice service will be automatically started every time you turn on your computer.
   ```