Voice Recognition Assistant

A Python-based voice assistant that listens for wake words, captures voice commands, and sends them to an n8n webhook for processing.

Features

- **Wake Word Detection**: Default wake word is "jarvis" (configurable)
- **Audio Acknowledgment**: Plays a custom audio file (WAV, MP3, etc.) when wake word is detected
- **Stop Phrase Detection**: Multiple stop phrases to end the session
- **n8n Webhook Integration**: Sends transcribed commands to your n8n workflow
- **Online Speech Recognition**: Uses Google Web Speech API
- **Optional Offline Wake Word**: Snowboy integration for offline wake word detection
- **Noise Reduction**: Optional audio processing with pydub
- **Cross-Platform**: Runs on Raspberry Pi, Linux, macOS, and Windows
- **Configurable**: Easy configuration via YAML file

Quick Start

1. Prerequisites

- Python 3.7+
- Conda (Miniconda or Anaconda)
- Microphone access
- Internet connection for speech recognition

2. Installation

```bash
# Clone or download the files to your project directory
# Run the setup script
chmod +x setup.sh
./setup.sh
```

3. Manual Installation (if setup script fails)

```bash
```

```
# Create conda environment
conda env create -f environment.yml

# Activate environment
conda activate voice-assistant

# Install system dependencies
# Linux:
sudo apt-get install portaudio19-dev python3-pyaudio

# macOS:
brew install portaudio

# Windows: Usually no additional steps needed
```

4. Configuration

Edit `config.yaml` to customize your settings:

```yaml
wake_word: "jarvis"
webhook_url: "https://n8n.casa-bakewell.com/webhook/discord-general-channel"
client_id: "voice_assistant"
```

5. Usage

```bash
# Activate the conda environment
conda activate voice-assistant

# Run the assistant
python3 voice_assistant.py
```

How It Works

1. **Wake Word Detection**: The assistant continuously listens for the wake word
2. **Audio Acknowledgment**: Plays a custom audio file to confirm wake word detection
3. **Command Capture**: After the tone, it listens for your command
4. **Speech Recognition**: Converts speech to text using Google's API

5. **Webhook Delivery**: Sends the transcribed command to your n8n webhook

Webhook Payload

The assistant sends JSON data to your n8n webhook in this format:

```json
{
  "body": {
    "content": {
      "text": "turn on the lights",
      "timestamp": 1642765432.123,
      "client_id": "voice_assistant"
    }
  }
}
```

Configuration Options

Setting	Description	Default
wake_word	Word to activate the assistant	"jarvis"
stop_phrases	Phrases to stop the assistant	["stop listening", "goodbye", "exit"]
webhook_url	Your n8n webhook URL	Required
client_id	Identifier for webhook payload	"voice_assistant"
recognition_timeout	Command timeout in seconds	5
energy_threshold	Microphone sensitivity	300
acknowledgment_tone.enabled	Enable wake word acknowledgment sound	true
acknowledgment_tone.audio_file	Path to audio file for acknowledgment	"acknowledgment.wav"
acknowledgment_tone.volume	Volume multiplier (0.0 to 1.0)	0.5
acknowledgment_tone.fallback_tone.frequency	Fallback tone frequency in Hz	800
acknowledgment_tone.fallback_tone.duration	Fallback tone duration in seconds	0.2
logging_level	Log verbosity	"INFO"

Troubleshooting

Microphone Issues

```bash
# Test microphone access
python3 -c "
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as source:
    print('Microphone working!')
"
```

Audio Dependencies

Linux:

```bash
sudo apt-get install portaudio19-dev python3-pyaudio alsa-utils
```

macOS:

```bash
brew install portaudio
```

Windows:

- Usually works out of the box
- If issues occur, try installing Microsoft Visual C++ Build Tools

Common Issues

1. **"No module named 'pyaudio'"**
   - Ensure you're in the conda environment: `conda activate voice-assistant`
   - Reinstall: `pip install pyaudio`
2. **"Microphone not found"**
   - Check microphone permissions
   - Test with system audio settings
   - Try different microphone index in code
3. **Recognition not working**

- Check internet connection (required for Google API)
- Verify microphone input levels
- Adjust `energy_threshold` in config

Advanced Features

Snowboy Integration (Optional)

For offline wake word detection:

1. Download Snowboy models from Snowboy
2. Place `.pmdl` file in project directory
3. Update `config.yaml`:

```yaml
snowboy_model: "your_model.pmdl"
```

Acknowledgment Sound

The assistant can play a custom audio file when the wake word is detected. Configure in `config.yaml`:

```yaml
acknowledgment_tone:
  enabled: true                # Enable/disable the sound
  audio_file: "acknowledgment.wav"  # Path to your audio file
  volume: 0.5                  # Volume multiplier (0.0-1.0)
  fallback_tone:               # Used if audio file fails
    frequency: 800             # Backup tone frequency
    duration: 0.2              # Backup tone duration
```

Supported Audio Formats:

- **WAV** - Recommended for best compatibility
- **MP3** - Popular compressed format
- **FLAC** - High-quality lossless format
- **OGG** - Open-source compressed format
- **Other formats** supported by soundfile library

Audio File Guidelines:

- **Length**: 0.1-1.0 seconds recommended for quick acknowledgment
- **Sample Rate**: Any (automatically handled)
- **Channels**: Mono or stereo (automatically converted to mono)
- **Quality**: Any bitrate/quality (higher quality = larger file)

File Examples:

```yaml
# Different format examples
audio_file: "chime.wav"          # WAV file
audio_file: "beep.mp3"           # MP3 file
audio_file: "notification.flac"  # FLAC file
audio_file: "/path/to/sound.ogg"  # Full path to OGG file
```

Creating Acknowledgment Sounds:

- Use short, pleasant sounds like chimes, beeps, or notification sounds
- Avoid long or jarring sounds that might interrupt your speech
- Free sounds available at freesound.org, zapsplat.com, or create your own
- Convert between formats using tools like Audacity, ffmpeg, or online converters

Backward Compatibility:

- Old `wav_file` configuration key still works
- Automatically detects and plays the correct format based on file extension

Fallback Behavior:

- If the audio file is not found, automatically falls back to a generated tone
- If both fail, logs a warning but the assistant continues working
- Supports graceful degradation for maximum reliability

Noise Reduction

Enable noise reduction in `config.yaml`:

```yaml
enable_noise_reduction: true
```

Requires pydub: `pip install pydub`

Development

Project Structure

```
voice-assistant/
├── voice_assistant.py     # Main application
├── config.yaml            # Configuration file
├── environment.yml        # Conda environment
├── requirements.txt       # Python dependencies
├── setup.sh               # Setup script
├── README.md              # Documentation
└── voice_assistant.log    # Log file (created at runtime)
```

Extending Functionality

The assistant is designed to be easily extensible:

- Add custom wake word detection methods
- Implement additional audio processing
- Create custom webhook payloads
- Add support for multiple n8n workflows

License

MIT License - feel free to modify and distribute.

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly
5. Submit a pull request

Support

For issues and questions:

1. Check the troubleshooting section

2. Review the logs in `voice_assistant.log`
3. Test with minimal configuration
4. Ensure all dependencies are properly installed