Befehle hinzufügen

Anleitung zum Hinzufügen eigener Befehle zum Voice Assistant.

Befehls-Typen

Der Voice Assistant unterstützt 3 Typen von Befehlen:

1. **Pattern-based Commands** - Einfache Keyword-Erkennung
2. **LLM-based Commands** - Intelligente Verarbeitung mit ChatGPT
3. **Home Assistant Commands** - Smart Home Steuerung

Pattern-based Commands

Wo hinzufügen?

Bearbeite `15_voice_assistant_configurable.py`, Funktion `execute_command()`.

Beispiel: Neuer Web-Befehl

```
def execute_command(command_text):
    text = command_text.lower().strip()

    # NEUER BEFEHL: Öffne Reddit
    if "reddit" in text:
        speak("Öffne Reddit")
        webbrowser.open("https://www.reddit.com")
        return True

    # ... (restliche Befehle)
```

Beispiel: Neuer App-Befehl (Windows)

```
# NEUER BEFEHL: Öffne Paint
if "paint" in text or "malen" in text:
    speak("Öffne Paint")
    subprocess.Popen("mspaint.exe")
    return True
```

Beispiel: Neuer System-Befehl

```
# NEUER BEFEHL: Wochentag
if "wochentag" in text or "welcher tag" in text:
    from datetime import datetime
    weekdays = ["Montag", "Dienstag", "Mittwoch", "Donnerstag",
                "Freitag", "Samstag", "Sonntag"]
    today = weekdays[datetime.now().weekday()]
    speak(f"Heute ist {today}")
    return True
```

Erweiterte Pattern-Matching

Regex für flexible Befehle

```
import re

# BEFEHL: "Setze Timer auf X Minuten"
match = re.search(r'timer.*?(\d+)\s*minute', text)
if match:
    minutes = int(match.group(1))
    speak(f"Timer auf {minutes} Minuten gesetzt")

    # Timer starten
    import threading
    def timer_callback():
        time.sleep(minutes * 60)
        speak("Timer abgelaufen!")

    threading.Thread(target=timer_callback).start()
    return True
```

Parameter-Extraktion

```
# BEFEHL: "Suche nach X auf Google"
if "suche" in text and "google" in text:
    # Extrahiere Suchbegriff
    search_term = text.replace("suche", "").replace("nach", "").replace("auf
google", "").strip()

    if search_term:
        speak(f"Suche nach {search_term}")
        webbrowser.open(f"https://www.google.com/search?q={search_term}")
        return True
```

LLM-based Commands

Automatische LLM-Weiterleitung

Wenn kein Pattern-based Command matched, wird automatisch das LLM verwendet (falls konfiguriert).

Eigene LLM-Prompts

Bearbeite `16_llm_integration_prototype.py`, Klasse `LLMManager`:

```
class LLMManager:
    def __init__(self, ...):
        # ANPASSEN: System-Prompt
        self.system_prompt = """Du bist ein hilfreicher Voice Assistant.

        Spezielle Regeln:
        - Antworte immer auf Deutsch
        - Halte Antworten kurz (max 2 Sätze)
        - Bei Mathematik: Zeige Rechnung
        - Bei Übersetzungen: Nur Ergebnis, keine Erklärung
        """
```

Home Assistant Commands

Voraussetzung

Home Assistant muss konfiguriert sein (siehe Home Assistant Integration).

Neues Gerät hinzufügen

```
def parse_home_command(text: str, ha_manager: HomeAssistantManager) -> bool:
    text_lower = text.lower()

    # NEUES GERÄT: Ventilator
    if "ventilator" in text_lower:
        entity_id = "fan.ventilator"

        if "an" in text_lower or "ein" in text_lower:
            ha_manager.call_service("fan", "turn_on", entity_id)
            speak("Ventilator eingeschaltet")
            return True
        elif "aus" in text_lower:
            ha_manager.call_service("fan", "turn_off", entity_id)
            speak("Ventilator ausgeschaltet")
            return True

    # ... (restliche Geräte)
```

Mehrere Räume

```
# BEFEHL: "Licht im [RAUM] an/aus"
rooms = {
    "wohnzimmer": "light.wohnzimmer",
    "schlafzimmer": "light.schlafzimmer",
    "küche": "light.kueche",
    "bad": "light.bad"
}

for room_name, entity_id in rooms.items():
    if room_name in text_lower and "licht" in text_lower:
        if "an" in text_lower:
            ha_manager.light_on(entity_id)
            speak(f"Licht im {room_name.capitalize()} eingeschaltet")
            return True
        elif "aus" in text_lower:
            ha_manager.light_off(entity_id)
            speak(f"Licht im {room_name.capitalize()} ausgeschaltet")
            return True
```

Befehls-Priorität

Befehle werden in folgender Reihenfolge verarbeitet:

1. **Höflichkeits-Befehle** (Danke, Hallo, etc.)
2. **Home Assistant Commands** (falls aktiviert)
3. **Pattern-based Commands** (Web, Apps, System)
4. **LLM-based Commands** (Fallback)

Priorität ändern

Ändere Reihenfolge in `execute_command()`:

```
def execute_command(command_text):
    # 1. Höflichkeit (immer zuerst)
    if text in ["danke", ...]:
        return True

    # 2. NEUE PRIORITÄT: Eigene Custom-Befehle
    if my_custom_command(text):
        return True

    # 3. Home Assistant
    if parse_home_command(text, ha):
        return True

    # 4. Pattern-based
    if pattern_commands(text):
        return True

    # 5. LLM (Fallback)
    return llm_command(text)
```

Beispiele: Nützliche Befehle

1. Wetter-Befehl (mit API)

```
import requests

def get_weather(city="Berlin"):
    """Holt Wetter von OpenWeatherMap API."""
    api_key = os.getenv('OPENWEATHER_API_KEY')
    url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid=
{api_key}&units=metric&lang=de"

    response = requests.get(url)
    data = response.json()

    temp = data['main']['temp']
    description = data['weather'][0]['description']

    return f"In {city} sind es {temp:.1f} Grad und {description}"

# In execute_command():
if "wetter" in text:
    weather = get_weather()
    speak(weather)
    return True
```

2. Musik-Befehl (Spotify)

```
import spotipy
from spotipy.oauth2 import SpotifyOAuth

def play_spotify(track_name):
    """Spielt Track auf Spotify."""
    sp = spotipy.Spotify(auth_manager=SpotifyOAuth(
        client_id=os.getenv('SPOTIFY_CLIENT_ID'),
        client_secret=os.getenv('SPOTIFY_CLIENT_SECRET'),
        redirect_uri="http://localhost:8888/callback",
        scope="user-modify-playback-state"
    ))

    # Suche Track
    results = sp.search(q=track_name, limit=1)
    track_uri = results['tracks']['items'][0]['uri']

    # Spiele ab
    sp.start_playback(uris=[track_uri])

# In execute_command():
if "spiele" in text and "musik" in text:
    # Extrahiere Song-Name
    song = text.replace("spiele", "").replace("musik", "").strip()
    play_spotify(song)
    speak(f"Spiele {song}")
    return True
```

3. Reminder-Befehl

```
import threading
from datetime import datetime, timedelta

def set_reminder(message, minutes):
    """Setzt Erinnerung."""
    def reminder_callback():
        time.sleep(minutes * 60)
        speak(f"Erinnerung: {message}")

    threading.Thread(target=reminder_callback).start()

# In execute_command():
match = re.search(r'erinnere mich in (\d+) minuten an (.+)', text)
if match:
    minutes = int(match.group(1))
    message = match.group(2)

    set_reminder(message, minutes)
    speak(f"Okay, ich erinnere dich in {minutes} Minuten")
    return True
```

4. Screenshot-Befehl

```
from PIL import ImageGrab
from datetime import datetime

def take_screenshot():
    """Macht Screenshot."""
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"screenshot_{timestamp}.png"

    screenshot = ImageGrab.grab()
    screenshot.save(filename)

    return filename

# In execute_command():
if "screenshot" in text or "bildschirmfoto" in text:
    filename = take_screenshot()
    speak(f"Screenshot gespeichert als {filename}")
    return True
```

Testing

Test-Skript erstellen

```python
# test_commands.py
from voice_assistant import execute_command

test_cases = [
    "Öffne YouTube",
    "Wie spät ist es?",
    "Licht im Wohnzimmer an",
    "Was ist 15 mal 23?",
    "Setze Timer auf 5 Minuten"
]

for test in test_cases:
    print(f"\nTest: {test}")
    result = execute_command(test)
    print(f"Result: {result}")
```

Debugging

Aktiviere Debug-Mode in `config.ini`:

```ini
[Advanced]
debug_mode = true
```

Dann siehst du alle Befehls-Verarbeitungen:

```
🔍 [COMMAND] Verarbeite: 'öffne youtube'
🌐 Öffne YouTube
✅ Befehl erfolgreich
```

Best Practices

1. Spezifische Keywords verwenden

❌ Schlecht:

```
if "musik" in text:  # Zu allgemein
```

✅ Gut:

```
if "spiele musik" in text or "musik abspielen" in text:
```

2. Fehlerbehandlung

```
try:
    result = execute_api_call()
    speak(result)
except Exception as e:
    print(f"Fehler: {e}")
    speak("Entschuldigung, das hat nicht funktioniert")
```

3. Feedback geben

```
# Immer TTS-Feedback
speak("Öffne YouTube")  # Bestätigung
webbrowser.open("https://youtube.com")
```

4. Dokumentieren

```
def my_custom_command(text):
    """
    Führt Custom-Befehl aus.

    Trigger: "mach X"
    Beispiel: "mach das Licht an"
    """
    # ...
```

Nächste Schritte

1. ✅ Füge eigene Befehle hinzu
2. ✅ Teste mit `test_commands.py`
3. ✅ Aktiviere Debug-Mode
4. ✅ Teile deine Befehle (Pull Request!)

Fragen? GitHub Discussions