Spracherkennung "Vosk" installieren und testen

Standalone / Python

Homepage https://alphacephei.com/vosk/ → Python-Paket installieren mit pip

sudo apt install ffmpeg build-essential python3-pip

Damit wir uns nicht die Installation von anderen KI-Tools (torch, janus, deepseek....) zerschießen, installieren wir alle Dependencies in ein separates Verzeichnis, das wird dann als PYTHONPATH angegeben beim Starten!

pip install --target venvs vosk

Beim Aufruf wird dann entweder jedes Kommando mit

PYTHONPATH=venvs Kommando...

vorangestellt, oder (gilt dann immer in dieser laufenden Shell)

export PYTHONPATH=venvs

Man kann den PYTHONPATH auch mit mehreren Verzeichnissen angeben, diese werden dann in der Reihenfoolge der Verzeichnisse durchsucht:

PYTHONPATH=venvs:/noch/ein/Pfad:. ...

Der . ist das aktuelle Verzeichnis.

Python-Programm für Live-Spracherkennung (Mikrofon als Eingabe, Empfehlung Konferenzmikro/-lautsprecher Jabra!):

```
#!/usr/bin/env python3

import subprocess
import re
from modules import aux_modules

from vosk import Model, KaldiRecognizer, SetLogLevel

SAMPLE_RATE = 16000

SetLogLevel(0)

# model = Model(lang="en")
model = Model(lang="de")
rec = KaldiRecognizer(model, SAMPLE_RATE)
start,start_a = 0,0
# input_device = 'plughw:1,0'
input_device = 'default'
phrase = ''
accumulating = False
# wake word hey photo is often confused with a photo by vosk...
wake_word_re = '^(hey|a) photo'
```

```
with subprocess.Popen(["ffmpeg","-loglevel", "quiet","-f","alsa","-i",
                           input_device,
                           "-ar", str(SAMPLE_RATE) , "-ac", "1", "-f", "s16le", "-"],
                           stdout=subprocess.PIPE) as process:

    while True:
        data = process.stdout.read(4000)
        if len(data) == 0:
            break
        if rec.AcceptWaveform(data):
            print('in first part')
            result = rec.Result()
            print(rec.Result())
            # print("Text[0]: ",result[0], "Text[1]: ", result[1], "Text[2]: ",
result[2])
            text = rec.PartialResult()
            start,start_a,accumulating,phrase =
aux_modules.process_text(wake_word_re,text,start,start_a,accumulating,phrase)
        else:
            print('in else part')
            result = rec.PartialResult()
            print(rec.PartialResult())
            # print("Text[0]: ",result[0], "Text[1]: ", result[1], "Text[2]: ",
result[2])
            text = rec.PartialResult()
            start,start_a,accumulating,phrase =
aux_modules.process_text(wake_word_re,text,start,start_a,accumulating,phrase)

# we never get here.
    print(rec.FinalResult())
    print('In final part')
    text = rec.FinalResult()
```

Die Datei modules/aux_modules.py liegt auf dem Server als Zip-Archiv, hier der Inhalt (ins Verzeichnis von Vosk als modules/aux_modules.py kopieren).

```
import re,time,json

def process_text(wake_word_re,text_s,start,start_a,accumulating,phrase):
    max = 5.5 # seconds
    inactivity = 10 # seconds
    short_max = 1.5 # seconds
    elapsed = 0
    if time.time() - start_a < inactivity:
# Allow some time to elapse since we just took an action
        return start,start_a,accumulating,phrase
# convert text to real text. Real text is in 'partial'
    text_d = json.loads(text_s)
    text = ''
    if 'partial' in text_d:
        text = text_d['partial']
    if 'text' in text_d:
        text = text_d['text']
    if not text == '': phrase = text
    if re.search(wake_word_re,text):
        if not accumulating:
            start = time.time()
            accumulating = True
            print('Wake word detected. Now accumulating text.')
    l = len(re.split(r'\s',text))
    print('text, word ct',text,l)
    if accumulating:
        elapsed = time.time() - start
        print('Elapsed time:',elapsed)
        if l > 1:
           phrase = text
    if elapsed > max or (elapsed > short_max and l == 1):
# we're at a natural ending here...
        print('This is the total text',phrase)
# do some action
```

```
# reset everything
        accumulating = False
        phrase = ''
        start_a = time.time()
    return start,start_a,accumulating,phrase
```

Homeassistant

In Einstellungen / Add-Ons zunächst das Repository `https://github.com/rhasspy/hassio-addons` unter Add-ons Store → Hambuergermenü rechts oben → Repositiries hinzufügen.

Ein Neustart von Homeassistent kann notwendig sein, damit der Inhalt aktualisiert wird.

Anschließend ist unter "Einstellungen" → Add-Ons ein "Vosk"-Plugin verfügbar, das installiert und konfiguriert werden kann (z.B. Sprache auf "de" umstellen).

Anschließend kann im HomeAssistant Sprachassistent (wieder Einstellungen→Sprachassistent) für Sprache-zu-Text das "vosk" ausgewählt werden.