Speech Interaction

– Dokumentation –

Mycroft - Kalender Skil

25.01.2021

Luca Krämer - lk152

Vincent Truong - vt026

0. Theorie

Mycroft ist ein Open source Sprachassistent, welchen man auf Linux Systemen oder auf dem Raspberry Pi installieren und vom User mit weiteren Skills erweitern kann. Skills werden in Python geschrieben und bieten dem System weitere Funktionalitäten.

Weitere Informationen zum Voice User Interface Design, sowie zum Aufbau und zu den verwendeten Technologien, findet man auf der MyCroft AI Website. Bevor ein Skill selbst geschrieben wird, ist es ratsam die Dokumentation "Skill Developement" zu lesen, um grundlegende Informationen zu bekommen und wichtige Begrifflichkeiten zu klären. Gibt es Probleme beim einrichten des Sprachassistenten, so bietet die Mycroft AI Website ein recht ausführliches Troubleshooting

Wake word: Wortfolge, die den Sprachassistenten aktivieren (z.B. Hey Mycroft)
Utterance: (übersetzt: Äußerung) des Uses die nach dem Wake Word gesprochen wird. (z.B. What time is it?)
Dialog: Ein Dialog ist eine dem Skill entsprechende Antwort der von Mycroft ausgegeben wird.
Intent: Ein Wort das aus einer Utterance erkannt und zu einem bestimmten Skill geteilt wird.

Beispiel:

```
User:     Hey Mycroft, what time is it?
Mycroft:  half past nine
```

1. Aufgabe

Die Aufgabe ist es, über den Sprachassistenten auf den eigenen Nextcloud Kalender zugreifen zu können. Das Ziel ist es zunächst, seinen nächsten Termin abzufragen. Mycroft sollte dementsprechend auf die Frage "Hey Mycroft, what's my next appointment?" mit dem nächsten Termin im Kalender antworten.

Beispiel:

```
User:     Hey Mycroft, what is my next appointment?
Mycroft:  Your next appointment is on June 22, 2020 at 4pm and is
          entitled Speech Interaction class.
```

2. Mycroft einrichten

Mycroft kann man entweder auf Linux installieren indem man das Repository mycroft-core klont und dann über das Terminal startet, oder man flasht die SD-Karte des Raspberry Pi's mit dem Picroft Image, was dazu führt, dass der Raspberry Pi direkt in die Konsole des Mycroft Systems bootet. Wir haben uns für das installieren von Picroft entschieden und verwenden ssh um auf die Konsole zuzugreifen. Anschließen gilt es das Device mit einem Mycroft AI Konto zu verknüpfen und um dort gewisse Dinge wie das Wake Word, oder die Zeit einzustellen.

2.1. Voraussetzungen

GitHub Account:	Wird benötigt um sich damit einen Mycroft Account zu erstellen und dort das Device zu konfigurieren und eine Übersicht der Skills zu bekommen. Ebenso werden für jeden Skill Git Repositories erstellt.
Raspberry Pi 4:	SD Karte mit dem Mycroft Image flashen und einrichten → genaue Anleitung und mehr Informationen in der Dokumentation
Python Kenntnisse:	Skills werden in Python geschrieben

3. Skill erstellen

- Skill erstellen

```
mycroft-msk create
```

- Dialog für Konfiguration folgen (siehe Doku)
- Git Hub verbinden und repository erstellen: manage-appointments-skill

4. Caldav installieren

Das Calendar Plugin wird benötigt um auf den Nextcloud Kalender zugreifen zu können. Mit dem Befehl

```
pip install caldav
```

wird das Modul installiert. Nun kann man das Modul in seine Python Datei importieren und so auf bestimmte Funktionen zugreifen.

5. Manage-appointments-skill

5.1. Datei __init__.py

- basic Skill
- imports

5.1.1. Basic Skill

```
from mycroft import MycroftSkill, intent_file_handler, intent_handler


class ManageAppointments(MycroftSkill):

    def __init__(self):

        MycroftSkill.__init__(self)

    @intent_file_handler('appointments.manage.intent')
    def handle_appointments_manage(self, message):

        self.speak_dialog("your next appointment is ...")


def create_skill():
    return ManageAppointments()
```

So sieht die __init__.py ungefähr nach dem Erstellen eines Skills aus. Eine genauere Beschreibung und Erklärung der Funktionen finden Sie in der Mycroft Dokumentation.

5.1.2. Imports

```
from mycroft import MycroftSkill, intent_file_handler, intent_handler
from datetime import datetime
import sys
import caldav
from caldav.elements import dav
```

Für das Verwenden des Caldav Plugins und den Zugriff auf den Nextcloud Kalender muss das caldav Modul importiert werden. Dabei ist wichtig das Modul auf dem Device mit

```
pip install caldav
```

zu installieren. Außerdem ist das Importieren des datetime Moduls notwendig für das parsen und sortieren der Events eines Kalenders.

5.2. getNextAppointment()

5.2.1. Beispieldialog

Es gibt zwei Möglichkeiten den nächsten Termin abzufragen:

```
What is my next appointment?
>> Your next appointment is on January
   25, 2021 at 8:30 AM and is entitled
   speech interaction
```

```
appointment
>> Your next appointment is on January
   25, 2021 at 8:30 AM and is entitled
   speech interaction
```

5.2.2. Implementierung handle_appointments_manage()

```
@intent_file_handler('appointments.manage.intent')
def handle_appointments_manage(self, message):
    '''handle_appointments_manage(self, message)

    The answer you receive for the question: What is my next appointment?
    Returned by calling getNextAppointment()

    '''
    self.speak_dialog(self.getNextAppointment())
```

Die Antwort auf die Frage “What is my next appointment? ” wird als String der Methode speak_dialog() übergeben sobald der intent erkannt wird. Um die ganze Antwort als String zu erhalten wird getNextAppointment() aufgerufen.

5.2.3. Implementierung getNextAppointment()

```
def getNextAppointment(self):
    '''getNextAppointment(self)

    return the string of the answer for the next appointment.


    '''

    # Events werden vom heutigen Tag bis 01.01.2024 abgerufen und sortiert
    events_fetched = self.loadEvents(datetime.today().year,datetime.today().month,datetime.today().day,2024,1,1)
    events_fetched.sort(key=lambda x: x.instance.vevent.dtstart.value.strftime("%Y-%m-%d"))


    if len(events_fetched)!=0:
        events_allDay = []
        events_notAllDay = []
        for i in events_fetched:
            if i.instance.vevent.dtstart.value.strftime("%H:%M") == "00:00":
                events_allDay.append(i)
            else:
                events_notAllDay.append(i)
        events_notAllDay.sort(key=lambda x: x.instance.vevent.dtstart.value)
        events_allDay.sort(key=lambda x: x.instance.vevent.dtstart.value)

        if len(events_notAllDay) == 0 :
            name = events_allDay[0].instance.vevent.summary.value
            date = events_allDay[0].instance.vevent.dtstart.value.strftime("%B %d, %Y")
            time = " all day long"

            return "Your next appointment is on " + date + time + " and is entitled " + name
        elif len(events_allDay) == 0 :
            name = events_notAllDay[0].instance.vevent.summary.value
            date = events_notAllDay[0].instance.vevent.dtstart.value.strftime("%B %d, %Y")
            time = " at " + str(events_notAllDay[0].instance.vevent.dtstart.value.hour+1) + events_notAllDay[0].instance.vevent.dtstart.value.strftime(":%M %p")

            return "Your next appointment is on " + date + time + " and is entitled " + name
        else:
            firstAllDay = events_allDay[0].instance.vevent.dtstart.value.strftime("%Y-%m-%d")
            firstNotAllDay = events_notAllDay[0].instance.vevent.dtstart.value.strftime("%Y-%m-%d")

            print(firstAllDay == firstNotAllDay)
            print(firstAllDay < firstNotAllDay)
            print(firstAllDay > firstNotAllDay)

            if firstAllDay == firstNotAllDay or firstAllDay < firstNotAllDay:
                name = events_allDay[0].instance.vevent.summary.value
                date = events_allDay[0].instance.vevent.dtstart.value.strftime("%B %d, %Y")
                time = " all day long"
                return "Your next appointment is on " + date + time + " and is entitled " + name
            else:
                name = events_notAllDay[0].instance.vevent.summary.value
                date = events_notAllDay[0].instance.vevent.dtstart.value.strftime("%B %d, %Y")
                time = " at " + str(events_notAllDay[0].instance.vevent.dtstart.value.hour+1) + events_notAllDay[0].instance.vevent.dtstart.value.strftime(":%M %p")
                return "Your next appointment is on " + date + time + " and is entitled " + name

    else:
        return "There are no upcoming events"
```

getNextAppointment() gibt den ganzen Antwortsatz für das nächste Event zurück.
konkreter Ablauf:

1) speichern und sortieren aller Events vom heutigen Tag bis bestimmtes Datum
2) Abfrage: sind Events vorhanden?
    a) Wenn ja, trennen und sortieren von All Day Events und Events mit einer Zeitangabe, dann 3)
    b) return "There are no upcoming events"
3) Abfrage: Gibt es All Day Events? Gibt es Events mit Zeitangabe? Gibt es beides?
    a) Nächstes Event mit Zeitangabe zurückgeben, wenn keine All Day Events vorhanden
    b) Nächstes All Day Event zurückgeben, wenn kein Event mit Zeitangabe vorhanden
    c) Wenn es beides am selben Tag gibt wird der All Day Event zurückgegeben.

Für jeden Fall wird ein angepasster String zurückgegeben return

```
"Your next appointment is on " + date + time + " and is entitled " + name
```

5.3. getAppointmentOnDate()

5.3.1. Beispieldialog

Man kann einen Termin abfragen, indem man folgenden Satz sagt: What appointments do I have on the <day> of <month>. Je nachdem was für Termine es an diesem Tag gibt, antwortet das System unterschiedlich:

1. Es gibt keine Termine an dem abgefragten Tag

```
What appointments do I have on the
27th of January
>> On the 27 of January you have no
   appointments
```

2. Es gibt genau einen Termin:

```
What appointments do I have on the
28th of January
>> On the 28 of January you have the
   following appointment: Speech
   Interaction project
```

3. Es gibt mehrere Termine an diesem Tag:

```
What appointments do I have on the
28th of January
>> On the 28 of January you have the
   following appointments: Speech
   Interaction project, Speech
   Interaction 2,
```

5.3.2. Implementierung handle_date_search()

```python
@intent_handler('appointments.manage.date.intent')
def handle_date_search(self, message):
    '''handle_date_search(self, message)

    The answer you receive for the question:
    What appointments do I have on the {day} of {month}?
    Returned by calling getAppointmentsOnDate()

    '''
    day = self.convertOrdinalToCardinalNumber(message.data.get('day'))
    month = self.convertMonthToInt(message.data.get('month'))
    if (day is not None and month is not None):
        self.speak_dialog(self.getAppointmentsOnDate(int(day),int(month)))
    else:
        self.speak_dialog("Please tell me the day and the month")
```

Ebenso wie bei handle_appointments_manage wird hier die Antwort auf die Frage “What appointments do I have on the {day} of {month}? ” als String, der Methode speak_dialog() übergeben sobald der intent erkannt wird.
Dabei wird vom User der Tag und der Monat übergeben was in den Variablen day und month gespeichert wird. Falls kein Tag oder Monat erkannt wird, wird der User dazu aufgerufen diese zu übergeben --> siehe else Zweig. Ansonsten wird getAppointmentsOnDate() aufgerufen mit dem man die Antwort als String erhält.

5.3.3. Implementierung getAppointmentsOnDate

```
def getAppointmentsOnDate(self,day,month):

    year = datetime.today().year
    events_fetched = self.loadEvents(year,month,day,year,month,day)
    events_allDay = []
    events_notAllDay = []
    for i in events_fetched:
        if i.instance.vevent.dtstart.value.strftime("%H:%M") == "00:00":
            events_allDay.append(i)
        else:
            events_notAllDay.append(i)
    events_notAllDay.sort(key=lambda x: x.instance.vevent.dtstart.value)

    if events_fetched == 0:
        return "This is not a day!"
    else:
        if len(events_allDay) == 0 and len(events_notAllDay) == 0:
            return "On the " +str(day) + ". of " + self.convertIntToMonth(month) + " you have no appointments."
        elif (len(events_allDay) == 1 and len(events_notAllDay) == 0) or (len(events_allDay) == 0 and len(events_notAllDay) == 1):
            result= "On the " +str(day) + ". of " + self.convertIntToMonth(month) + " you have the following appointment: "

            if len(events_allDay) == 1:
                myEvent = events_allDay[0].instance.vevent
                appointent_name = myEvent.summary.value
                result = result + appointent_name
            else:
                myEvent = events_notAllDay[0].instance.vevent
                appointent_name = myEvent.summary.value
                result = result + appointent_name

            return result

        else:
            result= "On the " +str(day) + ". of " + self.convertIntToMonth(month) + " you have the following appointments: "

            for event in events_allDay:
                myEvents = event.instance.vevent
                appointent_name = myEvents.summary.value
                result = result + appointent_name  +", "

            for event in events_notAllDay:
                myEvents = event.instance.vevent
                appointent_name = myEvents.summary.value
                result = result + appointent_name  +", "

            return result
```

getAppointmentOnDate() gibt den Antwortsatz für die Events an einem bestimmten Tag zurück.
Konkreter Ablauf:

1) speichern und sortieren aller Events eines bestimmten Tages
2) trennen und sortieren von All Day Events und Events mit einer Zeitangabe
3) Abfrage: Korrektes Datum übergeben:
    a) Wenn nein: return "This is not a day!"
    b) Wenn ja: 4)
4) String erstellen und zurückgeben je nachdem ob und was für Events vorhanden sind.

5.4. createNewEvent()

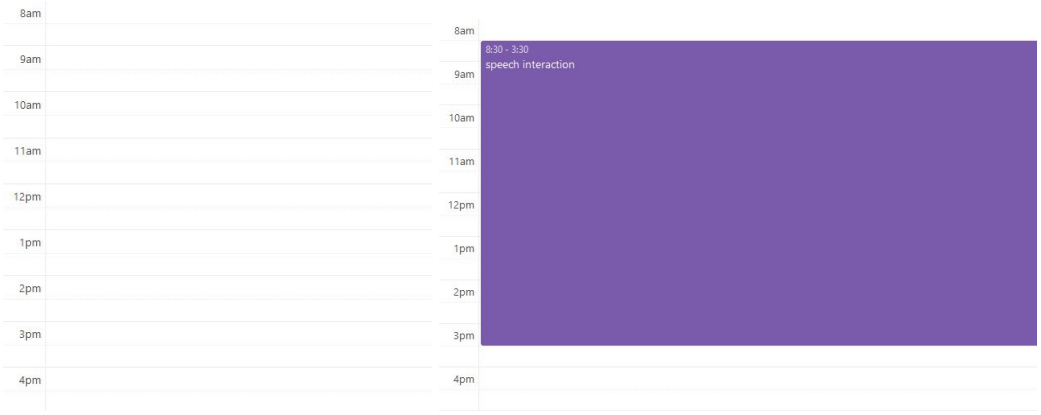
5.4.1. Beispieldialog

Es ist ebenfalls möglich über den Skill einen neuen Termin zu erstellen. Ein Beispieldialog zwischen dem User und dem Mycroft System könnte folgendermaßen aussehen:

```
create a new appointment on the 25th of january
>> how do you want to name the nex appointment?
speech interaction
>> When does the appointment start?
at 8:30
```

```
>> When does the appointment end?
15:30
>> The appointment speech interaction was succesfully created
```

Als Ergebnis wird ein neues Event im Kalender erstellt, mit den angegebenen Daten des Nutzers:



5.4.2. Implementierung Mycroft

Der Skill wird ausgelöst, wenn der Nutzer folgendes sagt: Create a new appointment on the <day> of <month>. Der Auslöser steht in unserer *appointments.manage.newEvent.intent* Datei. Sobald der Auslöser erkannt wird, startet die *handle_createNewEvent* Methode. Zuerst werden der Tag und der Monat als String herausgelesen und direkt in integer Werte umgewandelt. Diese Ints werden dann in den Variablen *day* und *month* gespeichert-

```
day = self.convertOrdinalToCardinalNumber(message.data.get('day'))
month = self.convertMonthToInt(message.data.get('month'))
```

Anschließend wird der Nutzer gefragt, wie er den Termin nennen möchte:

```
eventname = self.get_response('how do you want to name the nex appointment?')
```

Die Methode *get_response* startet das Mikrofon für 3-10 Sekunden und speichert dann den String in der Methode *eventname.*

Gleichermaßen werden dann vom Nutzer der Start und das Ende des neuen Termins abgefragt:

```
startDate = self.get_response('When does the appointment start?')
startHour = self.getHour(startDate)
startMin = self.getMinutes(startDate)

endDate = self.get_response('When does the appointment end?')
endHour = self.getHour(endDate)
endMin = self.getMinutes(endDate)
```

Die Uhrzeiten werden direkt in Integer umgewandelt und anschließend in den Variablen startHour, startDate, endHour und endDate gespeichert.

Die ganzen abgefragten Werte, werden dann in die Methode *createNewEvent* übergeben (siehe unten), in der dann der neue Termin erstellt wird:

```
self.createNewEvent(eventname,month,day,int(startHour),startMin,int(endHour),endMin)
self.speak_dialog("The appointment " + eventname + "  was succesfully created")
```

Am Ende bekommt der Nutzer noch die Rückmeldung, dass der neue Termine erfolgreich erstellt wurde.

5.4.3. Implemenmtierung createNewEvent()

Die *createNewEvent* Funktion wird aufgerufen, um ein neues Event im Kalender zu erstellen. Sie erwartet folgende Parameter:

```
def createNewEvent(self,name,month,day,startHour,startMin,endHour,endMin):
    """create a new event in the Calendar

     Parameters:
                name (string): the name of the new event
                month (int): the month of the new event
                day (int): the day of the new event
                startHour (int): the Hour of the startTime of the new Event
                startMin (int): the Minutes of the startTime of the new Event
                endHour (int): the Hour of the endTime of the new Event
                endMin (int): the Minutes of the endMinutes of the new Event
    """
```

Nachdem der Nutzer die Daten angegeben hat wird diese Methode aufgerufen (vgl. oben).

Zuerst wird eine Verbindung zum Kalender hergestellt:

```
url = "https://" + self.getUsername() + ":" + self.getPassword() + "@next.social-robot.info/nc/remote.php/dav"
client = caldav.DAVClient(url)
principal = client.principal()
# get all available calendars (for this user)
myCalendar = principal.calendars()[0]
```

Danach werden die Parameter so formatiert, dass sie in ein Kalenderobjekt eingefügt werden können.

```
DTcurr = datetime.today().strftime('%Y%m%dT%H%M%SZ')
UID = DTcurr = datetime.today().strftime('%Y%m%dT%H%M%SZ')
summary = name
start = datetime(datetime.today().year, month, day, startHour-1, startMin)
end = datetime(datetime.today().year, month, day, endHour-1, endMin)
DTstart =   start.strftime('%Y%m%dT%H%M%SZ')
DTend = end.strftime('%Y%m%dT%H%M%SZ')
```

Diese Variablen werden anschließend in das vorgefertigte Kalenderobjekt eingefügt:

```
        myNewEvent = """
BEGIN:VCALENDAR
VERSION:2.0
PRODID:ownCloud Calendar
BEGIN:VEVENT
CREATED;VALUE=DATE-TIME:""" + DTcurr + """
UID:""" + UID + """
LAST-MODIFIED;VALUE=DATE-TIME:""" + DTcurr + """
DTSTAMP;VALUE=DATE-TIME:""" + DTcurr + """
SUMMARY:""" + summary + """
DTSTART;VALUE=DATE-TIME:""" + DTstart + """
DTEND;VALUE=DATE-TIME:""" + DTend + """
CLASS:PUBLIC
END:VEVENT
END:VCALENDAR
 """
```

Das neue Event ist nun in der Variable *myNewEvent* gespeichert. Als letztes wird dann diese event im Kalender gespeichert:

```
myCalendar.save_event(myNewEvent)
```

6. Repository

Unser Skill ist in folgendem Repository zu finden:
https://github.com/vt026/manage-appointments-skill