**Speech-to-Text Node**

Der Speech-to-Text Node folgt einem linearen Konzept (wird in Paper, Poster und Artikel bei weitem deutlicher erklärt), wenn es um die Abarbeitung des Sprachbefehls geht. In diesem File wird primär über die einzelnen Teile dieses Nodes geschrieben und wie diese adaptiert werden können.

**Allgemeine Informationen**

Dependencies:

* Generell wegen der angedachten Funktionalität: ROS
* Panda & Husky haben nur eine ROS 1 Kompatibilität „out of the box“
* Panda benötigt Ubuntu 16.04 und einen Realtime-Kernel
* ROS 1 bietet nur Python 2.7 an
* Snips NLU benötigt Python
* Tensorflow kann nur auf Ubuntu mit Python 2.7 verwendet werden
* eSpeak hat einen Python-Wrapper

*Es ergibt sich somit folgender Set-Up: Ubuntu 16.04 mit ROS 1 Kinetic Kame und Python 2.7.*

WARNUNG: Der Support für Python 2.7 wird mit Anfang 2020 beendet! ROS 2 kann jedoch mit Python 3.X betrieben werden.

Durch die Verwendung von verschiedenen Tools, können nur begrenzt Sprachen verwendet werden. Die momentane Eingränzung findet durch Snips NLU (Intent & Variablen Detektion) statt. Es sind die geläufigsten Sprachen wie Deutsch, Englisch, Italienisch, etc. vorhanden. Eine komplette Auflistung kann unter folgendem Link gefunden werden: snips-nlu.readthedocs.io/en/latest/languages.html

Alle anderen Schirtte der Sprachsteuerung, die direkt mit Sprache zu tun haben (Speech-to-Text Transformation und Language Identifkation) sind abhängig davon, was der User benötigt. Genaueres in den jeweiligen Files.

**Sprachaufnahme**

Es wurde im Sinne der Privatsphäre die Sprachaufnahme so konzipiert, dass die Taste „R“ (für recording) gedrückt werden muss. Dies kann natürlich durch jede beliebige Taste ausgestauscht werden.

**Language Identification**

Hierbei handelt es sich um das trainierte Model, dass via pickle eingelesen wird, um anschließend die Prediktion zu vollziehen. Es wird nach dem höchsten Prozentsatz ausgewählt (German: 79% & English: 21% -> German).

Für weitere Informationen was das Training des Models betrifft bzw. die Feature Generierung bitte die dazugehörigen Files lesen.

**Intent & Variablen Detektion**

Die getCommand()-Funktion extrahiert den Namen des angesprochenen Roboters, den Intent des Befehls und wenn benötigt die jeweiligen Variablen. Snips NLU erlaubt es mehrere Intents auf einmal zu analysieren. Es muss hier nur darauf geachtet werden, dass der Output ein Softmax ist. Je mehr Intents in das Model eingespeißt werden, desto größer ist die Wahrscheinlichkeit, dass ein falscher Intent prognotiziert wird. Je nachdem welche Intent dann auch gewählt worden ist, muss die Variablenextraction variiert werden. Soll bedeuten, dass die Syntax für das Model von Intent zu Intent unterschiedlich ist (dementsprechend auch die Variablenbenenung) und somit auf andere Keys geachtet werden muss. Für eine genauere Erklärung bite das Snips NLU File ansehen.

**Command Publishing**

Das Command Publishing wurde so minimal wie nur möglich gehalten. In diesem Proof-of-Concept wird eine vordefinierte message mit jeweils benötigten Informationen gefüttert und anschließend auf ein einziges Topic gepublished.

**Text-to-Speech Node**

Im STT-Node werden auch Sprachbefehle generiert, die ausgegeben werden müssen/sollten. Da im Sinne der Konsistenz nur der TTS-Node diese Ausgaben machen soll, werden somit die jeweiligen Texte an den TTS-Node via Topic übergeben. Dank der Verwendung von eSpeak ist dies momentan nur ein Commandline Aufruf mit den dazugehörigen Parametern. Auch hier wird auf das zugehörige File verwiesen.