**Intent und Variablen Detektion**

Snips NLU wird für die extrahierung des Intents/Absicht und der dazugehörigen Variablen verwendet. Snips NLU muss zuvor ein Model trainieren, dass anschließend sprachenabhängig geladen werden muss. Somit muss für die korrekte ausführung pro Sprache mindestens ein Modell generiert werden. Für noch weiter Informationen wird die Snips NLU Webpage (snips-nlu.readthedoc.io/en/latest/index.html) empfohlen.

**Allgemeines**

Die Installation erfolgt via pip: *pip install snips-nlu*

Für etwaiige Sprachpackete (z.B.: Englisch) bitte folgenden Befehl ausführen: *python -m snips\_nlu download en*

**Modellgenerierung**

Pro Sprachen muss ein Modell erstellt werden. Dies kann entweder via einem YAML oder JSON File gemacht werden. Der Vorteil von YAML ist, dass es für den User leichter zu lesen ist. Da das Model jedoch nur JSON verarbeiten kann, muss das YAML anschließend transformiert werden. Im folgenden wird eine Anleitung für das Erstellen des YAML, die Transformation und Einspeißen in das Model gegeben.

Erstellen des YAML-Files:

Dieses File hat zwei wichtige Definitionsbereiche: Intent & Entity. Für einen visuellen Input bitte das Listing 1 konsultieren.

Im Falle des Intent muss hier definiert werden was für ein Name der Intent haben soll (wird beim der Verwendung des Models ausgegeben), welche Slots der Intent beinhaltet und Beispielsätze/Utterances. Bei den Slots der Intents kann weiter definiert werden, um was für einen exakten Slot es sich handelt. Als Beispiel kann die Entität Area betrachtet werden, die in Ziel\_Areal und Anfangs\_Areal unterteilt werden kann.

Nach dem Intent müssen noch die Entitäten/Slots der Intents definiert werden. Der Name gibt an worum es sich prinzipiell handelt (Objekt, Areal, etc.). Die Automatische Erweiterung/Extensible kann verwendet werden, wenn gewollt ist, dass das Model während dem Verwenden neue z.B.: Objekte berechnet. Diese werden jedoch nicht im sinne von Online-Learnin erneut in das Model gespeißt, sondern aus der Syntax des Satzes berechnet. Es kann somit auch möglich sein, dass ein komplett falsches z.B.: Objekt dadurch prognostiziert wird.

Die Values/Werte sind jene Werte, die das Model erlernt. Es besteht hier auch die Möglichkeit Synonyme zu definieren. Ein Beispiel wäre Panda bzw. Sam. Wird einer von beiden Namen erwähnt, wird das Model Sam ausgeben (da Panda als Synonym definiert worden ist). Bei Husky ist dem nicht so und es wird nur Husky ausgegeben.

# SO intent -> stands for Storage to Output: the desired item shold be transported from a Storage Slot to the Output

---

type: intent

name: SO

slots:

- name: object

entity: object

- name: origin\_area

entity: area

- name: goal\_area

entity: area

- name: robo\_name

entity: robo\_name

- name: storage\_slot

entity: storage\_slot

utterances:

- [robo\_name](Sam) bring das [object](Geschenk) vom [origin\_area](Lager) [storage\_slot](Fach nummer 1) zu der [goal\_area](Ausgabe).

- [robo\_name](Panda) gib mir den [object](Block) vom [origin\_area](Lager) [storage\_slot](Fach nummer 8).

- [robo\_name](Husky) [object](Eule) vom [origin\_area](Lager) [storage\_slot](Fach nummer 4) zu [goal\_area](Ausgabe).

- [robo\_name](Sam) [object](Geschenk) [origin\_area](Lager) [storage\_slot](Fach nummer 7).

# object entity

---

type: entity

name: object

automatically\_extensible: no

values:

- eule

- [geschenk, block]

# area entity

---

type: entity

name: area

automatically\_extensible: no

values:

- printer

- [storage, output]

# robo\_name entity

---

type: entity

name: robo\_name

automatically\_extensible: no

values:

- sam

- [panda] # This entry shows that panda is a synonym for sam

- [husky] # This entry shows that another name/robot (husky) exists, which is NOT a synonym for sam/panda.

# storage\_slot entity

# In the sample storage facility, which was used for the proof-of-concept, 9 slots were accessible. Therefore, the following entity definition was done by us 8 more times (1 - 8).

---

type: entity

name: storage\_slot

automatically\_extensible: no

values:

- 9

- [fach nummer neun, fach nummer 9, fach 9, fach neun, nummer 9, nummer neun, neun]

Listing 1: Beispielimplementierung eines YAML-Files für Snips NLU

Das YAML-File muss anschließend in ein JSON-File transformiert werden. Es kann folgender Befehl verwendet werden, wobei darauf geachtet werden muss, in welcher Sprache das YAML-File (Deutsch, Englisch, etc.) geschrieben worden ist: *snips-nlu generate-dataset de dataset.yaml > dataset.json*

Das Model wird nun via dem Code SnipsNLUTrain.py trainiert. Diese Option wurde absichtlich gewählt, da das Training, je nach länge des Datensatzes länger dauern kann.

**Verwenden**

Um nun das gespeicherte Model zu verwenden, muss nur mehr eine Zeile in den benötigten Code inkludiert werden (abgesehen von den Imports): *loaded\_engine = SnipsNLUEngine.from\_path(„path/to/directory“)*