Rasa NLU

deconcentrator

> Rasa is an open source solution for developing „AI assistants“ or chatbots. Rasa provides a stack consisting of the modules „Rasa NLU“ and „Rasa Core“. With the help of „Rasa NLU“ the user intention is determined from the received text message (Intent Recognition) and afterwards the NLU returns all intentions of the message sorted according to the „Confidence Score“. Training data is required to record the user's intentions. Furthermore, Rasa NLU allows „entity recognition“ to extract relevant terms from the text. The Rasa Core is a dialog engine that uses machine-learning trained models to decide which response to send to the user, such as greet the user. Furthermore, the core allows „session management“ as well as „context-handling“. Within the project only the component „Rasa NLU“ will be used, because only the functionality is needed to capture entities from a text message and to determine the user intention.

Table Of Content

Getting Started

The following instructions are intended to help the user run the Rasa-NLU-component on the local machine for development.

Understanding Rasa-NLU

Rasa NLU allows the processing of natural language to classify user intentions and extract entities from text.

e.g.

```
„Wie ist das Wetter übermorgen?“
```

The user intention is then determined from the text. In the figure below, the response to the message is displayed. The user intention („Wetter“) and the date for tomorrow are illustrated.

```
{
    "intent": {
        "name": "wetter",
        "confidence": 0.9518181086
    },
    "entities": [
        {
            "start": 19,
            "end": 29,
            "text": "übermorgen",
            "value": "2020-01-20T00:00:00.000+01:00",
            "confidence": 1.0,
            ...
        }
    ]
    ...
}
```

Training data is needed so that Rasa can identify the intention of a text. For this purpose, training data can be created in the form of Markdown or JSON. You can define this data in a single file or in multiple files in a directory.

To create a trained model for Rasa from the Markdown or JSON, Rasa offers a REST API. An alternative to creating trained models is to install Rasa on your local machine and then create the model using the command „rasa train nlu“. Rasa creates the training model (tar.gz) from the Markdown or JSON.

Furthermore Rasa NLU is configurable and is defined by pipelines. These pipelines define how the models are generated with the training data and which entities are extracted. For this, a preconfigured pipeline with „supervised_embeddings“ is used. „supervised_embeddings“ allows to tokenize any languages.

Perform Rasa locally

You need the local installation of Rasa to create and test training models. For this, you use the directory „training“.

Basic requirements

The following installations must be made:
- Pip
- Python (Version 3.6.8)
- Tensorflow
- Making further installations (Rasa Installation)
- If necessary, further installation via pip (depending on the message of the compiler)

Project structure

The following files and directories are important for the project to configure Rasa, customize training data and create appropriate training models.

- config.yaml:
contains the configuration of the NLU e.g. specification of the pipeline (how the trained model is generated)
- /data (directory):
contains training data in the form of JSON (Markdown would also be possible)
- /models (directory):
contains the trained model in the form of tar.gz.files The model is needed to capture entities and the user intent of a message.

Commands

You need to run the following commands in the directory '.training'.

```
# Create training-model:
rasa train nlu

# Communicating with Rasa NLU on the command line:
rasa shell nlu –m models/name-of-the-model.tar.gz
```

How to generate training datasets

In this project we write training data in the form of JSON, because JSON offers the possibility to extract entities from a text message. For this purpose the data was generated with the tool Tracy. In the image below, Tracy is shown with „Öffnungszeiten“. Entities are added as „slots“, such as „Ort“. Training data follows in the lower part of the picture. As training data, you can specify messages, which the user can send to the „chatbot“. Currently the three user intentions „Mensa“, „Wetter“ and „Öffnungszeiten“ are supported.

Add new Model for Rasa-Container (Docker)

You have to add the generated model (tar.gz) under the path „rasa-app-data\models“.

Use of Docker

You will need to install Docker in order to use the Docker-Compose-file for running the application.

Installation instructions for Docker

Installing

After the repository has been cloned and the prerequisites have been fulfilled, you can run the

Docker-Compose-file.

Fix Me! Fix Me! Fix Me!

```
# build and start Rasa-NLU-Container && serve at localhost:5005
docker-compose up

# stop and remove rasa-container, volumes, images and networks
docker-compose down

# do the same steps as "docker-compose down"
# additionally remove declared volumes in Docker-Compose-file
docker-compose down -v

# lists running containers
docker ps

# connect to the container with a bash
docker exec -it <Container-ID> bash
```

Overview

As part of the chatbot-project, microservices are supposed to run in Docker-Containers. In order to start several different services in containers at the same time, a Docker-Compose-File should be created. A Docker-Image is used for the Rasa NLU. Duckling has also been added as an Docker-Image for capturing date entries and allows to parse dates in a structured text.

```
version: '3.0'
services:
  rasa:
    image: rasa/rasa:1.6.0-spacy-de
    ports:
      - 5005:5005
    volumes:
      - ./rasa-app-data:/app
    command:
      - run
      - --enable-api
      - --cors
      - "*"
  duckling:
    image: rasa/duckling:0.1.6.2
    ports:
      - 8000:8000
```

The most important file for Rasa is the machine learning trained model (.tar.gz), which is written in the volume of the docker container. When executing the Rasa container, the model is needed to recognize user intentions.

HTTP-API

Rasa offers several REST APIs to provide server information, training models, etc. The Rasa features used in the project are listed here:

- **Serverinformation**:

You can query the Rasa-server whether it is still running or which Rasa version is available. You can also check which model Rasa is currently using.

- **Model**:

You can send requests via the Rest API of the Rasa server to create a trained model or load the model into Rasa. You can also send text to the server and Rasa will then determine the user's intention and the confidence score.

Links:

- HTTP-API (Retrieved 12.12.2019)
- OpenAPI-specification (Retrieved 12.12.2019)

Further Development

For further development, it is important that the existing training data be expanded and improved.

Further Reading

- Rasa Documentation (Retrieved 12.12.2019)
- Running Rasa with Docker (Retrieved 12.12.2019)

Built With

- Docker-Compose (Retrieved 12.12.2019)
- Docker Hub Rasa (Retrieved 12.12.2019)

Author

- **Abirathan Yogarajah**

References

- https://rasa.com/ (Retrieved 12.12.2019)
- https://botfriends.de/botwiki/rasa (Retrieved 12.12.2019)
- https://www.artificial-solutions.com/wp-content/uploads/chatbots-ebook-deutsche.pdf (Retrieved

12.12.2019)
- https://docs.docker.com/ (Retrieved 12.12.2019)