Deconcentrator-JS

> BeuthBot deconcentrator written in JavaScript

Feature

The deconcentrator uses different NLU processors to compare their results and tries to choose an best fitting answer. The NLU processors like RASA must know their domain on their own. The deconcentrator simply compares the confidence score of the intents given from the processors and returns the intent with the highest score.

Getting Started

Prerequisites

- node.js (Development)
- Docker (Development, Deployment)
- docker-compose (Development, Deployment)

Clone Repository

```
# clone project
$ git clone https://github.com/beuthbot/deconcentrator-js.git

# change into directory
$ cd deconcentrator-js

# copy environment file and edit properly
$ cp .env.sample .env
```

Install

There are two different ways running the deconcentrator. First is with `Node.js`'s package manager (`npm`) which is probably the better idea for developing. The other way is to run the deconcentrator-js in a Docker container with `docker-compose`.

Install with npm

Using npm you simply type in the following command.

```
# install dependencies
$ npm install

# start running the deconcentrator at localhost:8338
$ npm start run
```

This will run the deconcentrator on it's default port `8338` and with the default RASA service url `http://localhost:5005/model/parse`.

Install with docker-compose.yml

Using docker-compose is prossibly the easiest way of running the deconcentrator. Simply type

```
$ docker-compose up
```

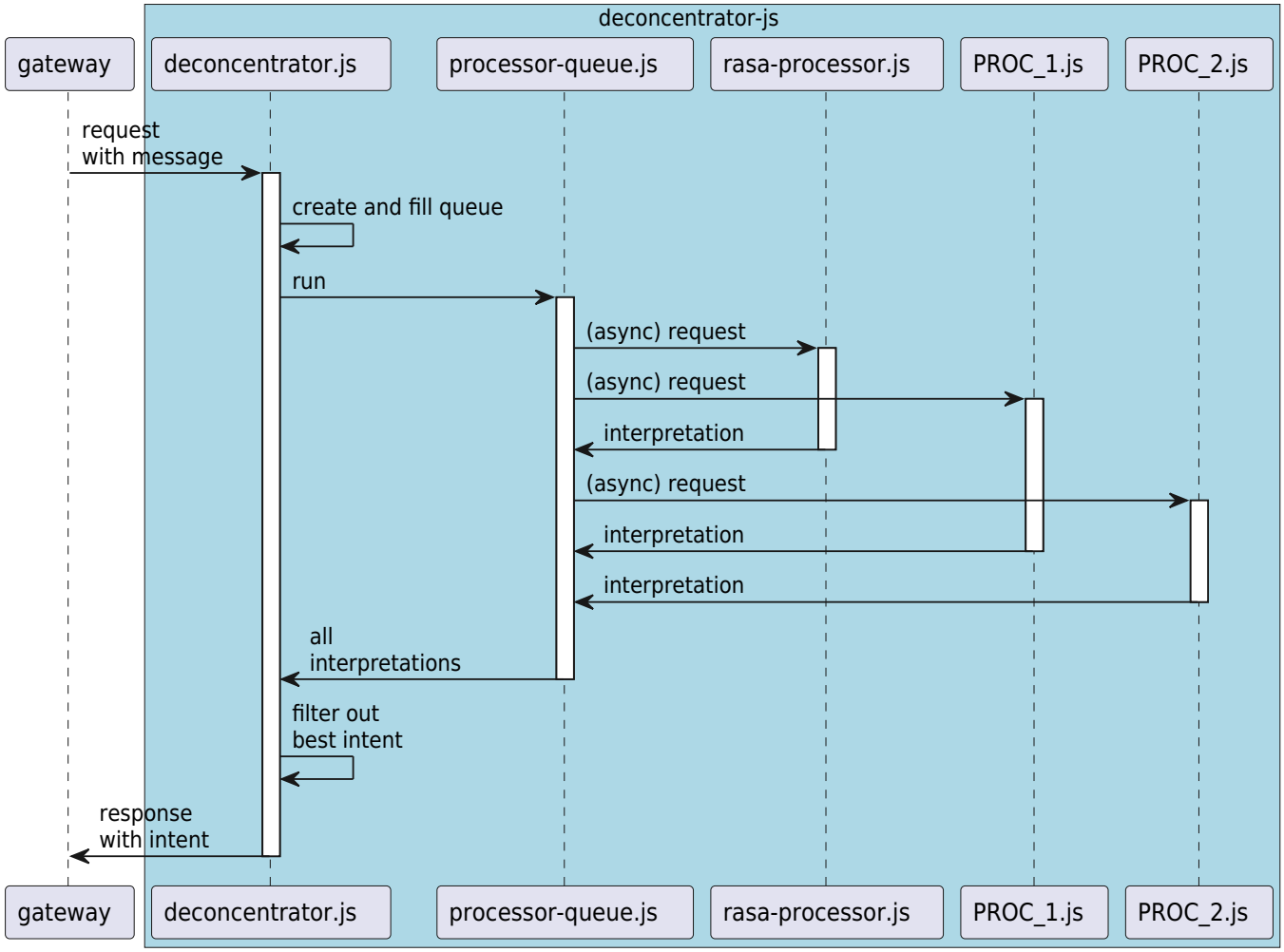
to run a container with the deconcentrator. The docker-compose file also uses port `8338` as a default one. The endpoint of RASA is taken from the `.env`. Make sure to edit it to your needs. Have a look at the sample file `.env.sample` and the section [.env](#.env).

Overview

Structure

Location	About
`.documentation/`	Contains documentation, UML and icon files.
`model/`	Contains the processors and the `processor-queue.js`.
`.env.sample`	Sample environment file.
`deconcentrator.js`	Main js file which is executed by Node.
`.gitignore`	Git ignore file.
`docker-compose.yml`	Defines the deconcentrator-js service.
`Dockerfile`	Defines the deconcentrator-js container.
`package.json`	Node project definition.
`README.md`	This document.

Functionality



deconcentrator.js

Uses an express application to listen for incoming messages. For an incoming message it then creates a processor-queue. The processor to can be specified with the `processors` property of an message. See [Request Schema - `Message`](Request-Schema —Message) for more information. After all processor are done with interpretation the result with a confidence score which is too low are filtered out.

processor-queue.js

For every incoming message the deconcentrator creates a new `ProcessorQueue` (defined in `processor-queue.js`) and adds all available processors to it. When calling the `.interpretate(message)` function of the queue it starts requesting the processors for an interpretation. The number of asynchronous requests can be set with the `numOfSynchronProcessors` property of the queue.

processor.js

Defines the interface of a NLU processor.

Implemented Processors

- rasa-processor.js
- ...

Add new NLU processor

Setp 1:

Create a new NLU processor service.

Step 2:

Create a new `PROCESSOR_NAME-processor.js` file in the `model` directory of the project.

Step 3:

Implement the `name` property and the `interpretate` function. Make sure the response looks like the one from rasa or the demo processor.

Step 4:

Add the name of the processor to the `default_processors` in the `deconcentrator.js` file.

API

The following lists the resources that can be requested with the deconcentrator API.

```
GET   http://localhost:8338
```

Returns a live sign of the deconcentrator.

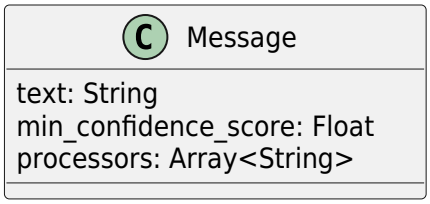
```
POST  http://localhost:8338/messages
```

Request Schema - `Message`

```
{
  "text": "Wie wird das Wetter morgen?",
  "min_confidence_score": 0.8,
  "processors": ["rasa"]
}
```

Whereas the specification of the `min_confidence_score` and the`processors` is optional. If not minimum confidence score is given a default one is used (by now this is `0.8`). For now there is only the usage of RASA implemented so there is no effect of specifying the `processors` property.

Class Diagramm



Response Schema - `Answer`

The response for a successfully processed request to the deconcentrator contains the following information.

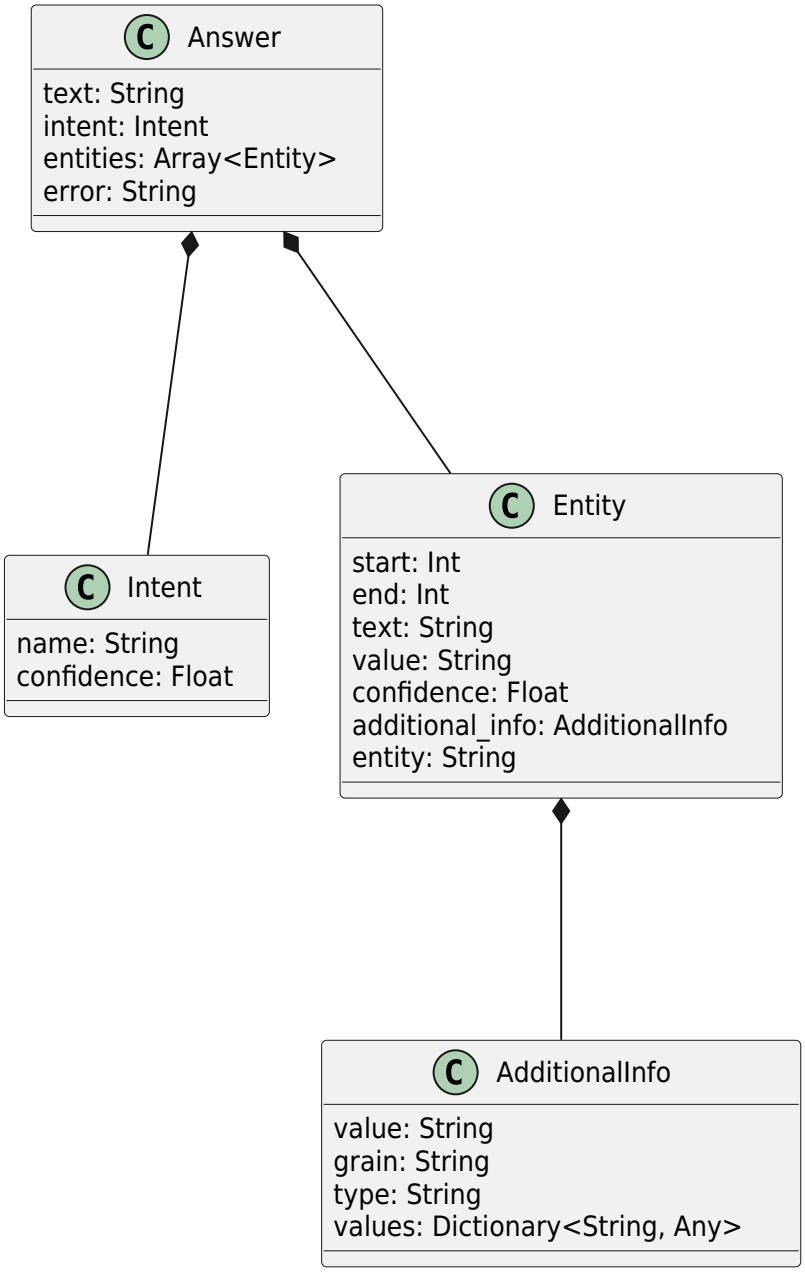
```
{
  "intent": {
    "name": "wetter",
    "confidence": 0.9518181086
  },
  "entities": [
    {
      "start": 20,
      "end": 26,
      "text": "morgen",
      "value": "2020-01-20T00:00:00.000+01:00",
      "confidence": 1.0,
      "additional_info": {
          "values": [
              {
                  "value": "2020-01-20T00:00:00.000+01:00",
                  "grain": "day",
                  "type": "value"
              }
          ],
          "value": "2020-01-20T00:00:00.000+01:00",
          "grain": "day",
          "type": "value"
      },
```

```
      "entity": "time"
    }
  ],
  "text": "Wie wird das Wetter morgen?"
}
```

The response for a unsuccessfully processed request to the deconcentrator or when an error occures contains the following information.

```
{
  "error": "The given message can't be interpretated.",
  "text": "Wie wird das Wetter morgen?"
}
```

Class Diagramm



Implemented and connected NLU processors

Provider	BeuthBot Project	Processor File
RASA	rasa	rasa-processor.js

More NLU processors candidates

- Microsoft LUIS - Google Cloud NLU - IBM Watson NLU

.env

With the `.env` file the deconcentrator can be configured. The following demonstrates a sample file. The same content can be found in the`.env.sample` file of the project.

```
RASA_ENDPOINT=http://0.0.0.0:5005/model/parse

# Optional
MIN_CONFIDENCE_SCORE=0.85
```

Requirements Analysis

- [x] `/DCF100/` The deconcentrator responds to incoming POST requests by delegating the message to a collection of NLU processor which try to interpretate the given message
- [x] `/DCF101/` The deconcentrator accepts incoming messages as defined via the Request Schema
- [x] `/DCF102/` The deconcentrator sends answers as defined via the Response Schema
- [x] `/DCF103/` The deconcentrator answers with proper messages for occuring errors
- [x] `/DCF104/` New NLU processors muss be easy to integrate
- [x] `/DCF105/` The deconcentrator has a default value for the minimum confidence score
- [x] `/DCF106/` The deconcentrator has a default value for the list of processors
- [x] `/DCF107/` The minimum confidence score can be set globally within the Dockerfile
- [ ] `/DCF108/` The list of processors to be used can be set globally within the Dockerfile