Raven Documentation

Release 0.1.1

Jawahar S

Jul 04, 2019

Contents:

CHAPTER 1

Raven

Raven is an umbrella package which is has intergation Raven-Bot(small NLU chatbot), TUI(simple Terminal User Interface) and Tx(Sanic web framework).

Warning: This project is still under hatching. So please support me till it hatches and spines of around the whole world, fast than speed of 'SuperMan'.

To know more about raven follow the document

Table of Contents

- *Step: 4*

CHAPTER 2

Contributing

Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change.

Please make sure to update tests as appropriate.

CHAPTER 3

License

MIT

3.1 Installation

CHAPTER 4

Stable release

This is the preferred method to install Tx, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

CHAPTER 5

From sources

The sources for Tx can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/jawahar273/Tx
```

Or download the tarball:

```
$ curl  -OL https://github.com/jawahar273/Tx/tarball/master
```

CHAPTER 6

Pipenv

Pipenv automatically create and manages the dependency using `pip`, `virtualenv` and adding/removing the packages using `Pipfile` and `Pipfile.lock`. One of the main object is remove the dependency on `requirement.txt` file.

```
pip install --user pipenv
```

The `requirements/*.txt` is compartable with Pipenv for recurrent installing of packages.

```
# step: 1 (for development/production)
pipenv install -r requirements/base.txt
# setp: 2 (for development only)
pipenv install -r requirements/local.txt --dev
# setp: 3 (only if you are need CLI)
pipenv install -r requirements/CLI.txt
# setp: 3 (only if you need Sanic)
pipenv install -r requirements/sanic.txt
```

6.1 To Get Starting

To get start with Raven-Bot might be difficult at first sight. Don't worry here are some instruction to get start with the bot. `Step:  1` is installing the piece of the code.

Warning: Hey if you thought the below section is hard to read then read this section *Usage*.

CHAPTER 7

Step: 2

Use of a CLI thing so called `scaff.py` which intensify generate scaffolding for us, which a half way through. There are two major sub division to remember in working with Raven-Bot one is `storage` is folder which contain of sub folder along with intents sets(`intents set` are collection of intent and organized in such a way for human classification) and other folder is called `response` which contain the code to executed if the engine choose the response to show.

Note: Please use *–help* before running the CLI to know what it is generating.

CHAPTER 8

Step: 3 (optional)

Human do love to interaction with people but with bot(who cares??? ah…). Who knows you may love it but not at first site. Through training might need regressively.

CHAPTER 9

Step: 4

Choose a nice folder for saving the dataset by using the `python scaff.py train` with proper argument you can successfully save the dataset under the default name ‘‘dataset.json‘‘(it can be changed with extra arguments).

CHAPTER 10

Usage

Warning: ..code-block:: bash

flask run –no-reload

Run the flask with auto reload disable.

10.1 Resourceful Command

Snips NLU has courpus dependency and this courpus can be diffrent language but in `Raven` we are currently concentrating only on english. This commnad is recommented if you are going start working raven bot.

```
python -m snips_nlu download en
```

10.1.1 tui.py

To run the bot in the TUI which can test or to run in production.

```
python tui.py
```

10.1.2 Scheduler

Huey is light weight task queue which schedule tasks to execute at a given time, or after a given delay, schedule recurring tasks, like a crontab, task prioritization, task pipelines etc..

Schedule mainter is main module to run separated along with Bot as is scheduler the task given by it. Huey will work along with the following dependency for flexibility such as RedisDB and along with Walrus a lightweight and hight level api Python utilities for working with redis-py.

```
huey_consumer maintainer.Huey -w 2 -c 2
```

Warning: Redis server must be running in background for huey to work properly.

Response schedule

Response schedule is base hub where all the scheduler inside the response must be importing here which intern it will be register with Huey producer for consumer. By using the parent, child approach the aggregation of the function can be identities by as a single point of entry in registering any response schedule.

10.2 Development

Development section will be used to describe the nessary step for bootstraping with `Raven`. After cloneing the repo the some of the sequenice of command are recommneded for handling some issue.

10.2.1 Pre-commit

Pre-commit is used to execute some function which can be defined in `.pre-commit-config.yaml`. To work around with the package following command are recommneded.

```
$ pre-commit install # Initializing pre-commit
    pre-commit installed at .git/hooks/pre-commit
$ pre-commit run # installing the pre-commit config packages.
    [INFO] Initializing environment for https://github.com/ambv/black.
    [INFO] Installing environment for https://github.com/ambv/black.
    [INFO] Once installed this environment will be reused.
    [INFO] This may take a few minutes...
    black................................................(no files to check)Skipped
```

10.2.2 merge_dev.py

10.3 Scaff/Generating Templates.

CHAPTER 11

Response Action Docs

11.1 Actions Working

Note:

Only the naming is the diffrent for avoiding the naming collision but actually it is refere to *raven.response*.

11.1.1 Default

Return the default resposne if the user's response is not understand or out of scope for the bot engine.

11.1.2 Github Package Version

Get the version of the given package based on the following verse. Thought there are some limitation in getting the version of the page as the api track set of package such as mysql, docker, etc.. To have better update follow the api github wiki

For example:

```
- check the version of this package redis and major
- will you check out for any update on this package docker in github
```

11.1.3 Humor

Bot are need to more friendly ah. Some-time we might me low on energy we might need joke to boost our self. This response return with some level of humber to user.

For example:

```
- tell me some jokes
- Do know any jokes
- I am sad today
```

11.1.4 Pwned

Do you know about data breach? If not then you should have atleast awareness of data breach. data breach is the intentional or unintentional release of secure or private/confidential information to an untrusted environment. Other terms for this phenomenon include unintentional information disclosure, data leak and also data spill.[1]

To check if your personal information has been breach in of the organization that you have been using in your day to day life or in the past. The response used Have I Been Pwned.[2]

For example:

```
- have my email id Claudine85@alo.com been hacked
- Check email id Nathanial112@hotmail.com been data breach
```

11.1.5 Reminder

Reminder response is used for setting a goal to be done with in a specific time.

Note: Right now the reminder response will be handling basic count down task.

Scheduler as the name suggest it schedule the given task with the help of Heuy a light weight task manager. Schedular are able to get the list of current running task(live one) and are able to revoke a task only if it is running or future task.

For example:

```
# count down only.
- countdown for 10 mins from now and Hello there.
- start a countdown for 20 mins after 20 mins and set task.

# get the list of countdowns
- show all countdowns list
- get me the countdown list and it id 12334499
```

[1] Accouding to Wikipedia
[2] Get more detail from the link

CHAPTER 12

Raven Bot Internal Structure

12.1 Architures

Raven-Bot compresseing of basic 5 components `raven.engine.abstract_engine`, `raven.input.abstract_input`, `raven.output.abstract_output`, `raven.layer.abstract_layer` and `raven.response.abstract_response`.

12.2 Environment

Name	Value	Describe
TARGET_PLATFORM	web, tui(default)	To Describe the current running platform which is web or TUI.
TEMPLATE_FORMATE	txt, html(default)	

CHAPTER 13

API

13.1 Raven Bot Component Api

13.1.1 Input

class `raven.input.abstract_input.`**`BaseInput`**(*txObject*)

`get_access_keys`()
Which return the current object access keys which will be used internally by input object class.

`processed`()
This method must be implemented on each new sub class. return must of dict type with *key*: 'PROCESSED_INPUT'.

class `raven.input.cli_input.`**`CLIInput`**(*txObject=None*)
CLIInput class is used to intergating with command line interface.

`processed`()
This method must be implemented on each new sub class. return must of dict type with *key*: 'PROCESSED_INPUT'.

`toBotText`(*text*)
Extra step for handling the input to the bot with custom method.

class `raven.input.rest_input.`**`RESTInput`**(*txObject*)
RESTInput is used to communication using REST operation.

`processed`()
This method must be implemented on each new sub class. return must of dict type with *key*: 'PROCESSED_INPUT'.

`toBotText`(*text*)
Extra step for handling the input to the bot with custom method.

13.1.2 Engine

class `raven.engine.abstract_engine.`**`AbstractEngine`**(**args*, ***kwarg*)

class `raven.engine.abstract_engine.`**`BaseEngine`**(*input_object*, *output_object*, *engine_param=None*)

Base engine will be mother of all its sub-class. As major lifting is taken in inside the *BaseEngine*. Just use the sub-class to handline the case which Mother's class can't be handling.

`go`()

Running the go function which will be used to run. Return is layer, will handling the input sentence.

..notes

```
sub class must return `self.return_object` itself.
Returning to ` OutPut` module must decised by ` Engine`.
```

`next`()

@depreacted :meth::*BasedEngine.next* are recommenced to be called after the end of engine.

`subscribe_tobreak`(*sender*)

Get the event signal and return true(for now).

class `raven.engine.default_engine.`**`DefaultEngine`**(*input_object*, *output_object*, *engine_param=None*)

Default Engine module will be used for managing the NLU engine.

`add`(*layer*)

Add the layer function into the engine's list for executing of the layer function.

Param `raven.layer.abstract_layer` layer the object

of function are added as layer into the engine for concurrency exection.

`command_success_response`(*txObject*)

Find if there the given user's text is related to command request. if then change the scope intent name as `commandsIntent_command`.

`go`(*pretty='base.html'*)

Param pretty is the name of the file where the meta data or base line of html are saved and it is parsed along with return result. Currently base.html and json.html is taken as parameter.

`parse`(*request_text*)

Parser the given user's text using the the Snip NLU engine.

`response`(*scope*)

Get the `Dict` status from NLU or command execution successfully, the one response class `raven.response` imported.

`train_model`(*path*)

Train the NLU JSON format by SNIP NLU.

Parameters **`path`** (*`str`*) – path of the dataset.json

13.1.3 Layer

class `raven.layer.abstract_layer.`**`AbstractLayer`**(**args*, ***kwargs*)

class `raven.layer.abstract_layer.`**`BaseLayer`**(*params*)

on_fails(*txObject*)
Wraper function for passing `response()` on the event of fails.

on_success(*txObject*)
Wraper function for passing `response()` on the event of success.

response(*txObject=None*)
response() must be defined by each sub class. Every subclass response must return *txObject* object

Command Request

class `raven.layer.cmd.cmd_base_layer.`**CMDBaseLayer**(*param=None*)

response(*txObject*)
response() must be defined by each sub class. Every subclass response must return *txObject* object

class `raven.layer.cmd.wiki.`**WikiLayer**(*param=None*)
Getting the Summary for the give keyword if it available in the wikipedia.

response(*txObject*)
response() must be defined by each sub class. Every subclass response must return *txObject* object

13.1.4 Ouput

Helper class that provides a standard way to create an ABC using inheritance.

13.2 Global Utils

Utils function for global usage.

`utils.`**env_bool**(*env_name: str*, *default: bool*) → bool
Get the environment variable's value convert into :Class:Boolearn

`utils.`**env_float**(*env_name: str*, *default: float*) → float
Get the environment variable's value convert into :Class:Float

`utils.`**env_int**(*env_name: str*, *default: int*) → int
Get the environment variable's value convert into :Classs:Init

`utils.`**env_str**(*env_name: str*, *default: str*) → str
Get the environment variable's value convert into string

`utils.`**import_class**(*path*)
Import the give class based on the given path

`utils.`**import_module**(*path*)
Import the give module based on the given path

`utils.`**render_template_file**(*file_name: str*, ***kwargs*)
Higher level api for rendering templates.

Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY

CHAPTER 14

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index