BSc in Computer Science

Snjallmennis ráðgjafi
Íba

Operational guide

Diemut Haberbusch
Helga Rún Steinarsdóttir
Laura Orsini R. Franca

December 2019

Instructor: Gunnar Sigurðsson

Examiner: Sigurjón Ingi Garðarsson

Client: Íslandsbanki

Table of contents

1. Introduction

This operational guide is for both maintenance as well as further development of the project. It goes through the steps needed to get the different services that compose the project to get all of them running and it explains their structure.

2. Rasa

2.1. Setup

1. Clone project from its repository to make it run locally from computer. For this, you must have the git CLI installed:

```
$ git clone https://github.com/Ibabot/Iba.git
```

2. Install Rasa Open Source with `pip`. The machine running it is required to have Python 3.6 or 3.7 previously installed.

```
$ pip install rasa
```

Make sure Microsoft VC++ Compiler is also installed. You can get it from the Visual Studio IDE.

This setup is originally made in the Windows 10 OS. If needed for another operating system, the Rasa official installation guide can be referred to. The Rasa documentation can also be used if something is not working during the installation.[1]

2.2. Development

1. Overview of the files in the project

   `actions/actions.py` - contains custom actions code to implement more complex logical operations and connections to external services and APIs.

   `actions/bank_data.json` - contains the JSON with the current information of branches of Íslandsbanki.

   `data/lookup` - contains lookup files to improve entity extraction.

   `data/nlu` - contains the training data used for the classifications of intents and extracting entities from user input.

   `data/stories` - contains the possible conversation paths used to train Rasa's dialogue management.

[1] https://rasa.com/docs/rasa/user-guide/installation/

`models/` - contains the trained models. The newest model is automatically used when running the bot.

`domain.yml` - contains the domain, i.e. the brain of the bot (it specifies the intents, entities, slots, and actions your bot should know about) as well as response templates

`config.yml` - training configurations for the NLU pipeline and policy ensemble.

`credentials.yml` - contains credentials for different services used and lists events to be captured and emitted by services that use it.

`endpoints.yml` - contains the webhook configuration for the custom actions.

2.2.1. Local Development without Docker

1. To start Íba, a trained model is needed That can be achieved by using the command listed below, using a terminal or the cmd prompt from the root of the project. This command must be performed every time changes are made to the code.

```
$ rasa train
```

2. To talk to the bot on localhost in cmd line interface, use the command below. The *--debug* flag is optional but it will set the server to be verbose during debug time.

```
$ rasa shell --debug
```

3. To use the custom actions, a separate terminal must be used. It is needed to run on another port

```
$ rasa run actions
```

2.2.2. Local Development with Docker

1. Once the download is complete, if using Windows, it is necessary that the OS version has Hyper-V support. More can be read about the issue here: https://docs.docker.com/machine/drivers/hyper-v/. If your OS version does support Hyper-V, make sure that the feature is enabled.
2. Docker must be installed. You can download it here.
3. The model must be trained through docker for it to work properly. Navigate to the project's root directory and type:

```
$ docker run -v $(pwd):/app rasa/rasa:1.5.1 train --domain domain.yml
--data data --out models
```

4. To run the server with docker, type:

```
$ docker run -v $(pwd)/models:/app/models rasa/rasa:1.5.1 run
```

5. To run the action server along with the core server, use the command:

```
docker-compose up
```

2.3. Deployment

1. When new code is pushed to the master on Ibabot's repository, the CI server (Heroku) automatically builds and deploys the project. If the build succeeds, the changes are pushed to the core Rasa server (https://ibachatbot.heroku.com) along with the Rasa Action server (https://rasibabot.heroku.com).

   Heroku will take in the Dockerfiles and assemble the images to run both servers (action and core). This process is done automatically, and can be edited in the `heroku.yml`.

3. Website

The website is developed in React v.16.11 and uses SCSS as for CSS preprocessing.

3.1. Setup

1. The website is a javascript application. As such, you must have *node.js* installed. You can do that by accessing this url.
2. Once node is installed, to download the project into your local machine, open your terminal and type:

```
$ git clone https://github.com/Ibabot/iba-website.git
```

   Note: Git CLI must be installed.

3. Navigate to the root of the project with the terminal and run the following command to install all dependencies:

```
$ npm install
```

4. Once the installation of the packages is complete, run the following to serve it:

```
$ npm start
```

It will open by default on port 3000.

3.2. Development and Deployment

The website is currently hosted on Heroku along with the Rasa backend. It can be found here: https://ibachatbot-webhook.herokuapp.com/

The project uses a widget developed by Botfront and is referred to in the official Rasa documentation, called Rasa Webchat (it can be found here). It is implemented as a component, called Chatbox.

All changes made and pushed to master will automatically build and deploy the website to production on the website's URL listed above.

4. Facebook

To integrate Íba with Facebook messenger it is needed to have a Facebook page to connect it to. The connection is done through the API & internal framework offered by Facebook for Developers.

1. Create messenger app in Facebook for Developers
2. Get a page-access-token from **Token Generation** in *Settings -> Access Tokens*.
3. Locate the secret from **App Secret** from *Settings -> Basic*
4. Add the previous information into the file `iba/credentials.yml` to set up the webhook endpoint from rasa's side. The *verify token* should be unique and it is decided by the user.

```
facebook:
  verify: "<verify token>"
  secret: "<App secret from Facebook>"
  page-access-token: "<Access token from Facebook>"
```

5. Set up a webhook from Facebook *Settings->Webhooks* by adding a callback URL (in Íba's case, here:https://ibachatbot.heroku.com/webhooks/facebook/webhook) and verify with chosen verify token.
6. By completing these steps, Íba should be connected to Facebook Messenger.