pybotframework Documentation

Release 0.1a1

Micheleen Harris, Kendall Chuang, David Clark

May 01, 2018

Contents

CHAPTER 1

User Guide

1.1 Introduction

Description: Python wrapper and package for Bot Framework REST API and State REST API (not production ready, yet).

The project,‘‘pybotframework‘‘, provides an easy way to create intelligent Python-based chatbots. It leverages Microsoft's Bot Framework REST APIs for easy deployment and connection to channels like Slack, Skype, FB Messenger and more. We aimed it to have rich dialogs by providing gateways to custom trained machine learning models and dialog logic.

It includes a great development experience due to its compatibility with an open source channel emulator. It also uses the familiar web microframework, Flask, for the web app component that can be customized later on. For authentication it leverages Flask-OIDC.

A snippet of a basic bot with custom ML:

```
# Labels for ML model
target_names = ['negative', 'positive']

# Instatiate the connector to custom ML model
sklearn_lang_conn = SklearnConnector(model_file='sentiment.pkl',
                                     target_names=target_names)

# Instatiate the bot
my_app = BotFramework(connectors=[sklearn_lang_conn])
```

Currently the project is Python 3 compatible, with Python 2.7 support pending demand.

1.2 Tutorials

We have the following tutorials available (as Jupyter notebooks):

Tutorial	Topics	Link
Regex Tutorial	regex, intents	Link 1
Sentiment Tutorial	scikit-learn, sentiment analysis	Link 2
TensorFlow Word2Vec Tutorial	TensorFlow, word2vec	Link 3
Microsoft LUIS Tutorial	LUIS Cognitive Service	Link 4

1.3 Examples

Examples are the actual example bots. Here is a listing and instructions on how to run them.

1.3.1 Bots Available

Example	Topics	Link
Eliza bot	regex json, response json	Link 1
Scikit-learn language bot	scikit-learn, sentiment analysis	Link 2
TensorFlow word2Vec bot	TensorFlow, word2vec	Link 3
Microsoft LUIS bot	LUIS Cognitive Service	Link 4

1.3.2 Running the Bots

This example is a Regex python bot using the Flask microframework to work with the Bot Framework on Azure or even just locally with the Bot Framework Emulator (recommended to use for testing and dev).

Setup

- Download the Bot Framework Emulator for local testing (https://github.com/Microsoft/BotFramework-Emulator#download) - multiple OS compatibility.

(For a full list of prerequisites see the main Readme on the repo here).

Run the Bot

From the bot's base folder and on the command line:

```
python <bot script name>.py
```

e.g.

```
python eliza_bot.py
```

This will start the flask server.

Test Locally in Emulator

1. Open up the BF Emulator (usually called *botframework-emulator* on your system)
2. Click on the "Enter your endpoint URL" and select or type in *http://localhost:3978/api/messages*
3. Leave the "Microsoft App ID" and "Microsoft App Password" blank

4. Click "CONNECT"

You should see in the Log window a "conversationUpdate" appear twice with no errors. If there's an error ensure you have the <*bot script name*>.*py* script running on the command line.

Deployment

Go to Deployment for deploy instructions.

1.4 Deploying Your Bot

1.4.1 Prerequisites

- Account on Docker Hub (Sign up at https://hub.docker.com/)
- Azure subscription (Sign up at https://azure.microsoft.com/en-us/free/)

1.4.2 Instructions

Linux Web App

1. From Azure Portal (portal.azure.com) spin up a Linux Web App
2. Log in to Docker locally (`docker login` and enter your credentials for Docker Hub)
3. Build the docker image from the bot's Dockerfile (you can name the image anything you wish, here it's `flaskbot`). For example:

```
docker build -f Dockerfile -t flaskbot:latest .
docker images
```

4. Tag the image with the Docker Hub username (replace <dockeruser> with your docker user name and give the image name) and tag:

```
docker tag <image id> <dockeruser>/flaskbot
```

5. Push up the docker image to Docker Hub (takes some time):

```
docker login
docker push <dockeruser>/flaskbot
```

6. Specify the docker image to the Linux Web App (give the app a name, replacing <mybotname>, resource group, replacing <mybotrg>, and point to the image):

```
az login
az webapp config container set --name <mybotname> --resource-group <mybotrg> --docker-
↪registry-server-url <dockeruser>/flaskbot
```

6. Navigate to portal and click on Docker Container in left panel to ensure the app is pulling in the correct image from Docker Hub.
7. Check that the endpoint is working:

- Navigate to http://<your web app name>.azurewebsites.net/api/messages and should get a Method Not Allowed http error (because GET not allowed here - but this message indicates the flask app is running)

Bot Framework Portal

Register your bot with the Microsoft Bot Framework: register.

Connect it to channels as is shown in connect.

Happy chatting!

CHAPTER 2

API Documentation

2.1 API Documentation

2.1.1 Bot Framework

2.1.2 Connector Classes

class pybotframework.connector.**Connector**

We use this as a public base class. Customized connectors inherit this class as a framework for building them.

Note: Base connector class class as framework for child classes.

respond(*message*)

This is called by botframework in child classes.

Parameters **message** (*str.*) – Cleaned message.

Returns str – the return code.

class pybotframework.regex_connector.**RegexConnector**(*intent_file*, *response_file*)

CHAPTER 3

Contributor Guide

3.1 Guide to Contributing

Please follow the contributing to open source docs here: http://contribution-guide-org.readthedocs.io/.

Useful links (more stuff coming soon):

- Issue Tracker: github.com/michhar/pybotframework/issues.
- Source Code: github.com/michhar/pybotframework.

Index

C

R