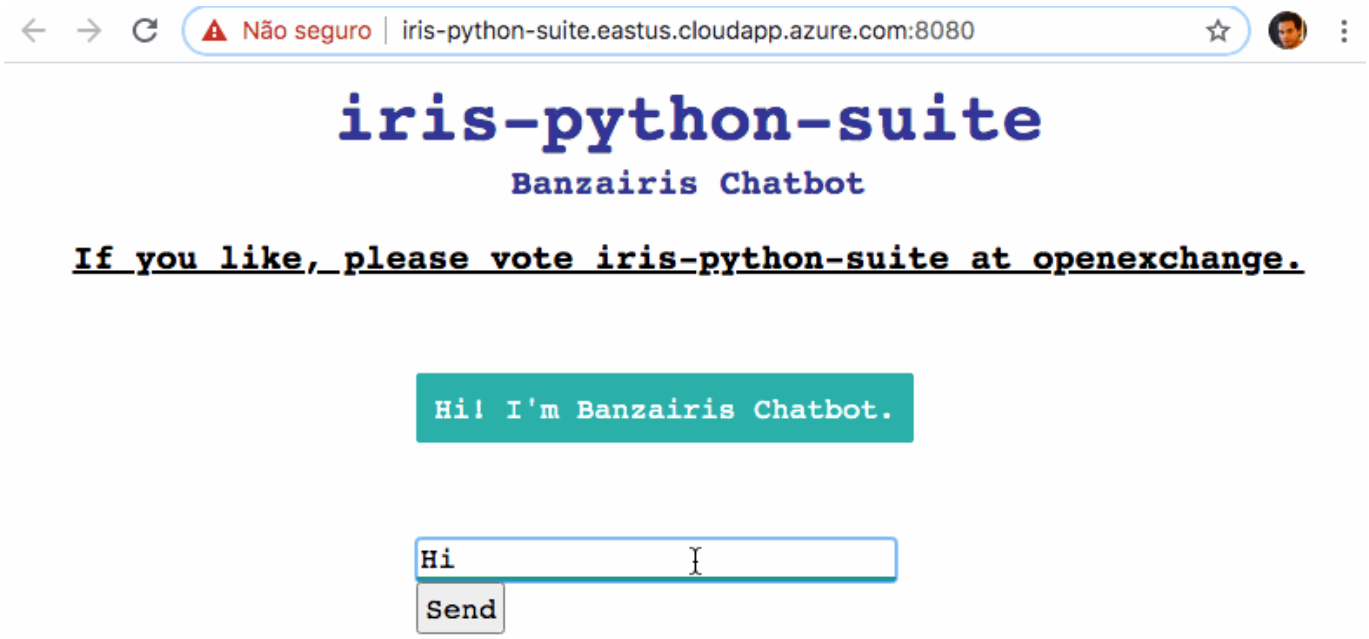
Article
Renato Banzai · Jun 3, 2020 4m read

Open Exchange

Creating a Chatbot with IRIS and Python



Creating a Chatbot with IRIS and Python

In this article I'm going to show how to integrate the InterSystems IRIS Database with Python to serve a Machine Learning Model of Natural Language Processing (NLP).

Why Python?

With a large adoption and use in the world, Python have a great community and a lot of accelerators ¦ libraries to deploy any kind of application.
If you are curious (https://www.python.org/about/apps/).

Iris Globals

As I start to learn about ^globals they became familiar to use as a fast way to ingest data in a out-of-box data model.
So at first I'm going to use ^globals to store training data and the conversations to log the chatbot behaviour.

Natural Language Processing

Natural Language Processing or NLP is a subject of AI that create the ability to read, understand meaning from our languages to machines. As you can imagine it's not quite simple but I'm going to show how perform your first steps in this
wide and beautiful field.

Demo - Try it yourself

I have deployed the Chatbot application as a demo here:
http://iris-python-suite.eastus.cloudapp.azure.com:8080

How does it work?

Machine Learning

First is good to know that Machine Learning has a different paradigm compared to common software development. The main point that is hard to understand is the cycle of development of machine learning models.

Shallow explanation alert

A standard application development cycle are like:

Develop Code->Test (with development data)->Deploy(use real data)

And Machine Learning the Code by itself don't has the same value. It shares the responsability with data! And no any data,
real data! Because the final code to be executed is generated by a merge between development concepts and the data used.
So a machine learning application cycle should be like:

Develop(Train) Model+Real Data->Validate->Deploy the result of this (a Model)

How to train a Model?

There are a lot of techniques to train models and each case and objective needs a big learning curve. In this case I use
the ChatterBot library that encapsulate some techniques and provide train
methods and pre-processed training data to helps us to focus on results.

Pre-trained Model Languages and Custom Model

You can start with this to have a basic conversational chatbot. You also can create all the data to train your chatbot,
that can be perfect to your needs but terrible to make in short time. I this project I use encorewebsm as the base of
conversation and merge with custom training data that you can create by a form

Basic Architecture

What did I use in Python

In this application environment I use Python 3.7 with these modules:

- PyYAML<=5.0.0
- dash==1.12.0
- dash-bootstrap-components==0.10.1
- dash-core-components==1.10.0
- dash-html-components==1.0.3
- dash-renderer==1.4.1
- dash-table==4.7.0
- plotly==4.7.1
- numpy==1.18.4
- networkx==2.4
- Flask>=1.0.0
- chatterbot>=1.0.0
- chatterbot-corpus>=1.2.0
- SQLAlchemy>=1.2
- ./nativeAPIwheel/irisnative-1.0.0-cp34-abi3-linuxx8664.whl

Project Structure

This project has a simple structure to be easy to understand. On the main folder we have 3 most important subfolders:

- ./app: with all the **application code** and installing configuration.
- ./iris: with the **InterSystems IRIS dockerfile** preparing to serve the application.
- ./data: To link the host to the container environment by a **volume**

Application Structure

Now inside the ./app directory we can see some files:

- chatbot.py : with the implementation of the web application
- irispythonsuite.py : a class with some accelerators to use with IRIS Database and Python by the IRIS Native API.

Database Structure

This application uses Intersystems IRIS as a repository, the globals used are:

- ^chatbot.training.data: stores all custom training data in the format of question and answers.
- ^chatbot.conversation : stores all conversation payload.
- ^chatbot.training.isupdated : controls the training pipeline.

Merchandise of my other solution

I didnt create a report to all conversations but it isnt a problem, with my global graph viewer I can follow the conversations.

Running the application by yourself

Prerequisites

- git
- docker and docker-compose (and more memory settings in docker at least 4GB)
- acess to a terminal in your environment

Steps

With docker-compose you can easily up one environment with all the pieces and configurations go to the iris-python-covid19
folder and type this:

```
$ docker compose build
$ docker compose up
```

Estimated time to up containers

1st time running will depend of your internet link to download the images and dependencies.
If it last more than 15 minutes probably something goes wrong feel free to communicate here.
After the 1st time running the next ones will perform better and take less then 2 minutes.

If is everything ok

After a while you can open your browser and go to the address:

The training data form

```
http://localhost:8050/chatbot-training-data
```

The chatbot

```
http://localhost:8080
```

You should look at IRIS Admin Portal

I'm using for now the USER namespace

```
http://localhost:9092
user: _SYSTEM
pass: theansweris42
```

If this article help you or you like the content vote:

This application is at the current contest on open exchange, you can vote in my application **iris-python-suite** here (https://openexchange.intersystems.com/contest/current)

#AI #Framework #Machine Learning #Python #InterSystems IRIS #Other
Check the related application on InterSystems Open Exchange