



QUICK START GUIDE

Jan 2018

Disclaimer -

The information contained in the document is confidential, privileged and only for the information of the intended recipient and may not be used, published or redistributed without the prior written consent

Contents

Quick Started Guide	2
Prerequisite	2
Installing python module	2
Deployment of hmi.jar	3
Sample Use Case – Trip Booking	4
Use Case – Trip booking	4
Browser Client	4
Sample Dialogue.....	5
Sample Rest APIs of Bot Engine.....	6
Step -1 Creating BOT instance.....	6
Step -2 Start conversation	6

Quick Started Guide

Below steps describe the deployment of HMI conversational UI (Bot) on Windows Machine.

Required configuration of Windows machine - Minimum 8 GB RAM with 64-bit OS with x-64 based processor

Prerequisite

1. Install Java 1.8+ and all system variables (JAVA_HOME etc.) are set
2. Download python-3.6.1.exe from <https://www.python.org/downloads/> (or latest python 3+ version)

Installing python module

1. Install scikit-learn
`pip3 install sklearn`
2. Install numpy module
Download "numpy-1.13.0+mkl-cp36-cp36m-win32.whl" from <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>
Copy it to d: folder and open command prompt in that folder. Run below command.
`# pip3 install numpy-1.13.0+mkl-cp36-cp36m-win32.whl --user`
3. Install scipy module
Download "scipy-0.19.1-cp36-cp36m-win32.whl" from <http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>
Copy it to d: folder and open command prompt in that folder. Run below command
`# pip3 install scipy-0.19.1-cp36-cp36m-win32.whl --user`
4. Install remaining packages
`pip3 install nltk`
5. Load the nltk data. Go to python prompt

```
>> import nltk  
  
>>nltk.download('punkt')  
  
>>nltk.download('stopwords')  
  
>>> nltk.download('averaged_perceptron_tagger')...(experimental)  
  
>>> nltk.download('wordnet')....(experimental)
```
6. Install flask module
`pip3 install flask`

Deployment of hmi.jar

This is conversation engine and following steps are required to deploy it.

1. Copy below mentioned folder structure from git repository to D:/HMI folder
2. Ensure that you have following folder structure

```
lib
res
WEB-INF
hmi.jar
```

3. Open command prompt, go to D:\HMI folder and run following command.

```
# java -jar hmi.jar -i rest -p 8080 -r trip =>(trip is sample dialog flow created for demo purpose)
```

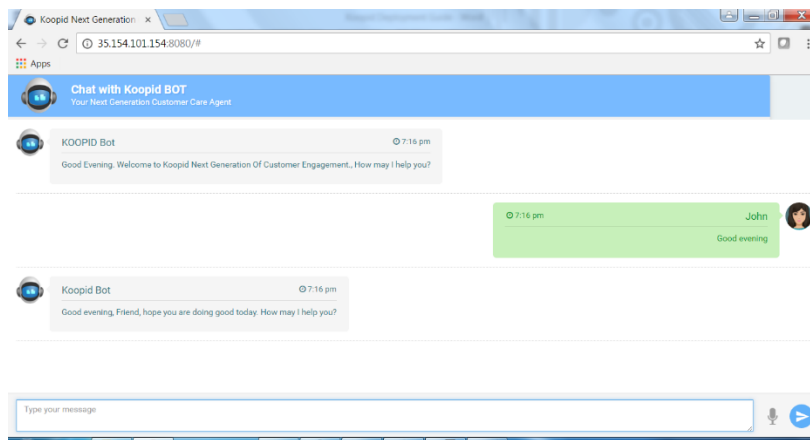
4. This will run conversation engine using inbuilt jetty server on port 8080.

Sample Use Case – Trip Booking

This use case demonstrates the bot's conversational capabilities. The use case is of user who wants to book a ticket or check weather information.

Use Case – Trip booking

A simple browser client that uses REST APIs is provided for testing this sample use case. Please note that this is just for demonstration purpose and does not represent the final solution.



Browser Client

Go to chrome browser and enter

URL - <https://<Your Machine IP>:8080>

- (accept the self signed certificate by clicking advanced

-> Proceed to <Domain name> (Unsafe)

- Following login screen would appear

Login to your account

John

.....

<https://35.154.101.154:8080>

Login

- Click on “IP” config button on right hand top side and enter your IP address with port (ex. <https://35.154.101.154:8080>)
- Enter user “John” and password “John123”
- You can login into the system
- You can use text window to CHAT or click MIC button for speech based conversation.

Sample Dialogue

B: Good evening, How may I help you?

U: Good evening

B: Good evening, Friend, hope you are doing good today. How may I help you?

U: I want to book a ticket

B: Please tell me what city do you want to travel?

U: I want to start from Atlanta

B: And for how many persons?

U: How is weather in Atlanta?

B: The temperature in Atlanta is 7.51 degrees. do you want to know about other cities?

U: no

B: for how many persons?

U: for 4 persons

B: When do you want to leave?

U: I want to leave on Sunday

B: which class do you want to travel?

U: Business

B: Hey John, this trip to Atlanta costs 255 Dollars. What can I do for you?

Sample Rest APIs of Bot Engine

The BOT engine has exposed different APIs so that it can be seamlessly integrated with different clients such as browser client, mobile apps, Messenger Apps and smart devices like Alexa and google home.

Following steps are necessary to initiate the conversation with BOT

Step -1 Creating BOT instance

One can create the BOT instance before user initiates a conversation. As an enterprise user you need to pass the user parameters to create the BOT instance. This is sent through url encoded body parameters. (user = John)

You can use any suitable REST client tool (curl, postman, soapUI etc.) to send below HTTP request. Enter below details to your RESTClient tool (postman or soapUI etc.)

Method – POST

URL - <http://35.154.101.154:8080/hmi/bot>

Param - url-encoded-body parameter – **key => user value => John**

e.g. (using curl)

```
# curl -k -i -X POST --data-urlencode "user=John" http://35.154.101.154:8080/hmi/bot
```

RESPONSE

HTTP/1.1 201 Created

Date: Mon, 09 Jan 2018 01:15:26 GMT

Location: **<http://35.154.101.154:8080/hmi/bot/d3-DXWQDQ8WDDFQ>**

Content-Type: text/plain

Transfer-Encoding: chunked

How may I help you?

Once this request is sent, Bot Server will create a unique URL for every user and returns it in HTTP header response. (As highlighted in RED above)

Step -2 Start conversation

Now that bot instance is created, you can start your conversation by sending your query through “userUtterance” parameter.

Important – check URL

Method – POST

URL - <http://35.154.101.154:8080/hmi/bot/d3-DXWQDQ8WDDFQ>

Param - url-encoded-body parameter – **key => userUtterance value => Good Morning**

e.g. (using curl)

```
# curl -k -i -X POST --data-urlencode "userUtterance=Good Morning"  
http://35.154.101.154:8080/hmi/bot/d3-DXWQDQ8WDDFQ
```

RESPONSE

HTTP/1.1 200 OK

Date: Mon, 09 Jan 2017 01:29:51 GMT

Content-Type: application/json

Transfer-Encoding: chunked

```
{  
  "user": "John",  
  "sessionId": "d2-QSKGDLBH7IJP",  
  "timeStamp": "2018-03-25 23:21:12",  
  "language": "en",  
  "source": "trip",  
  "result": {  
    "query": "Good Morning",  
    "reply": "Hello and hope you are good this morning. What can I do for  
you?",  
    "speech": "Hello and hope you are good this morning. What can I do for  
you?",  
    "intent": {  
      "name": "start",  
      "label": "Initial Task"  
    },  
    "currentEntity": {  
      "name": "welcome",  
      "label": "",  
      "type": "open_ended",  
      "value": ""  
    },  
    "entities": [  
      {  
        "name": "welcome",  
        "label": "",  
        "type": "open_ended",  
        "value": ""  
      }  
    ],  
    "contexts": []  
  },  
  "action": {  
    "intent": {
```



```

        "name": "",
        "label": ""
    },
    "entities": []
},
"iCard": {
    "type": "taskList",
    "tasks": [
        {
            "id": "1",
            "name": "getTripInformation",
            "label": "Book ticket"
        },
        {
            "id": "2",
            "name": "getWeatherInformation",
            "label": "Weather information"
        },
        {
            "id": "3",
            "name": "getWikipediaCityInfo",
            "label": "City information"
        },
        {
            "id": "4",
            "name": "cancelTask",
            "label": "Cancel"
        }
    ]
}
}

```

As can be seen the response shows the BOTs reply in JSON format.

Key JSON Objects –

reply – Bots reply to user query

intent – Identified intent

entities – entities that are part of intent

action – is filled once BOT takes any backend action with entities that are found through user conversation.

User can continue to converse by passing his/her query through new POST method and modifying “userUtterance” parameter.

e.g.

```
# curl -k -i -X POST --data-urlencode "userUtterance=How is weather in London"  
http://35.154.101.154:8080/hmi/bot/d3-DXWQDQ8WDDFQ
```

The list of other APIs are also available (available in manual document)

- terminate the instance of BOT
- Active dialogues that are being handled by BOT server
- Active tasks of specific dialogue instance
- Get conversation history of specific dialogue instance
- Store dialogue conversation in specific DB for further analysis and training