Northeastern ALIGN Program Chatbot Webhook

This brief manual contains some of the specifications for the webhook which helps resolve analytical and dynamic values for the Dialogflow chat bot. The manual will also give direction on what is the proper syntax that Dialogflow should provide when sending the answer to the hook in order to properly slot the correct values into the queried answer.

Specifications: The webhook is written in python. The web service that is attached to it which listens to requests uses the Flask web framework. All of the analytical data is stored within memory. However, in order to fetch the memory, it uses a Redis cache which is stored locally on another server. The source code has some sample functions in how data is grabbed and extrapolated within the cache.

Installing the webhook: To deploy the webhook you must have python 2.7 or better and the required dependencies which are outlined in requirements.txt in the source code directory. To do this make sure that pip is installed onto the machine. To install pip see: https://pip.pypa.io/en/stable/installing/

When pip is installed execute the following command in the working directory: 'pip install -r requirements.txt'

Running the webhook: To run the flask server type the following command in the working directory: 'python application.py'. Alternatively, if the flask command is available (via pip), you can set the $FLASK_APP environmental variable to the working directory then perform 'flask run'.

Under the hood: The following files provided in the source code are as shown:

- answers.json – Configuration file that has all of the statistical answers that is populated through the Redis cache
- application.py – The main configuration file which places all the necessary endpoints for the webhook to function
- config.json – The configuration file which holds key value pairs for Dialogflow and Backend credentials
- statistics.py – The helper script which preloads and resolves statistical values from the Redis cache which then populates into answers.json
- templates/ - The working directory which stores all of the views for the flask server. Currently it only has one webpage which serves as a standalone test shell for developers to interact with the Chatbot

Endpoints: Below are the provided REST endpoints for the Chatbot Webhook

Request	Endpoint	Description
GET	/	The standalone shell which can be opened up in a browser. Using a command line interface will simply give you the HTML source code
GET	/api?query=<Parameter>	The call in which you enter a query in <Parameter> and then the bot returns a response in JSON. *The query URI parameter is required*
GET	/api/intents	Retrieves all of the intents from the Dialogflow agent's API. Used for analytics, debugging, and developmental purposes.
POST	/webhook	The main webhook value which requires the JSON data structure from the current Dialogflow agent. The webhook will then refine the answer that came from the bot and return it to the bot agent.

Resolving answers from the webhook: When resolving answers from the webhook the placeholder value that's shown in the answers in Dialogflow must match the value that is reflected in answers.json. For example, if the count of students wants to be retrieves in an answer. Then the answer on Dialogflow should be #student_count# as opposed to the static answer that is shown. Make sure that it is spelled correctly with the proper casing or else the value will not resolve successfully.

Creating new dynamic values for the webhook: To create a new dynamic value to be resolved through the webhook, the JSON data structure for a statistical answer must be a shown:

```
"variable_name": {
    "fn": "function_to_resolve_to",
    "params": [
        "param"
    ],
    "value": ""
},
```

1. Set the name of the desired variable under “variable_name”. This will be the value that will be searched and resolved by the webhook POST request
2. fn – The function name within statistics.py to resolve to. To view a template of how the data should be grabbed and extrapolated see some of the sample code for the existing routes in statistics.py
3. params – If the function that is being resolved requires parameters this entry is a requirement. Resolve it with the proper parameters that should be used.
4. Value – Keep the value entry blank. This will be the value that gets resolved from the return type of the resolved function.

Dealing with configurations: Configurations for the application are placed within config.json. The config.json file has two configuration settings: one for Dialogflow and the other for direct access to ALIGN’s backend systems. To find the configurations settings for Dialogflow’s agent ID, key and development key. You must be the main administrator for the Dialogflow account. If you are the administrator for the Dialogflow account. Log into the console and on the side bar navigation click on the gear icon which is next to the agent’s name. The next panel that is shown will provide both the key, development key and the ID of the agent. To deal with the backend systems credentials (If changed), enter in the new username and password to access the backend systems. If you do not have the proper credentials contact the administrator who manages the backend systems for ALIGN.