Chat-Bot README

First Time Use Instructions

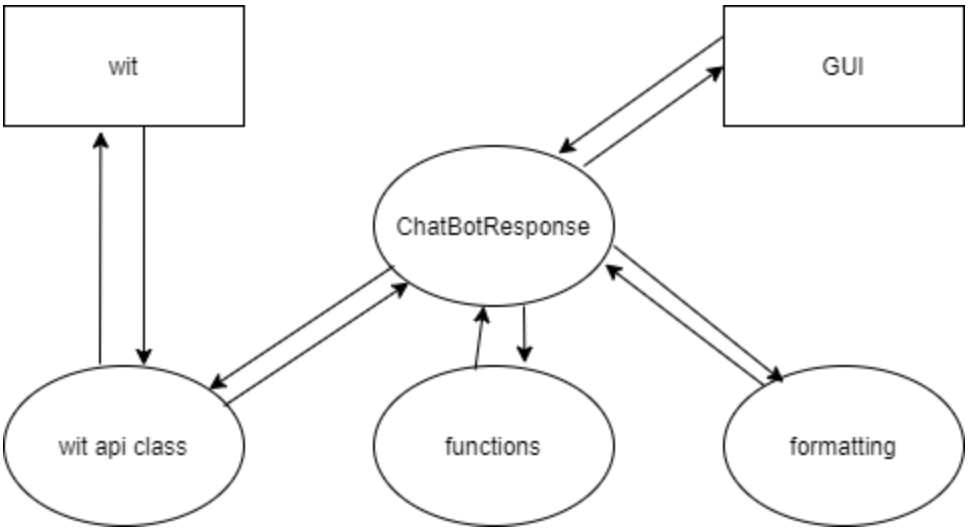
Our project relies on certain Python libraries for API calls and other functionality. We have included all the required libraries as a “requirements.txt” file. To use this you must run “pip install -r requirements.txt”. You will also need Python installed to run it.

To run the chatbot, download the zip file off of Github and extract the file. Open a console of any sort capable of running pip commands and navigate to the directory “chat-bot”, then run the command “pip install -r requirements.txt”. Once dependencies are downloaded, navigate to chat-bot, and run “main.py”. This will open a GUI where you can talk to the bot.

Project Description

Our team is developing an interactive conversational weather and navigation agent that will utilize various methods and APIs to respond to relevant user questions. The user can ask the bot questions regarding the current and future weather predictions and get brief POI information.The role of this bot would be to answer the frequent questions a user would need to know before leaving their house.The structure of our project is to make use of a third party software (wit.ai) to train and develop an existing AI that will help identify specific traits, entities and intents within the user question; then to use those parameters to query data via weather and geographic data APIs and provide output via a GUI.

Class Organization



Chat-Bot README

Class Responsibility/Description

location.py
Responsible for all location and distance functions.

getLocation(place = None)
Input: Place or location as a string, can be any format (city, country, address, etc)
Output: Tuple containing lat and long coords as float values of the location. If no location is given, it will default to using the user's location. Return type is a tuple of floats.

distanceByLatLong(entities)
Input: List of entities containing 1-2 locations
Output: Distance between the 2 locations in KM, if only 1 location is given, the second location is the location of the user. Return type is float.

timechatbot.py
Responsible for all timezone and time functions.

getTimezone(entities, place = None)
Input: List of entities containing locations, or a location as a string.
Output: If a string is given for 'place', outputs the timezone of that location. Otherwise outputs the timezone of the first location entity in the entity list. Return type is string.

getLocalTime(entities, place = None)
Input: List of entities containing locations, or a location as a string.
Output: If a string is given for 'place', outputs the local time of that location. Otherwise outputs the local time of the first location entity in the entity list. Return type is datetime.

getTimeDifference(entities)
Input: List of entities containing 1-2 locations.
Output: If 2 locations are input, get the difference in the 2 local times. If 1 location is given, it assumes the second location is the user's location. Return type is datetime.

witapi.py
Responsible for making API requests to wit.ai.

sendRequest(question)
Input: String to be sent to the chatbot.
Output: JSON containing entities, intents, and traits from wit.ai. Return type is JSON.

Chat-Bot README

responsechatbot.py
Responsible for handling all function calling based on the response from wit.ai to a question.

getResponse(question)
Input: String to be sent to the chatbot.
Output: String that is the chatbot's response. Return type is string.

responseformat.py
Every function in responseformat.py takes in the return data from their respective function from other classes, as well as all the inputs that were sent to them. They all return a formatted string to make the bot's talking sound better.

Example:
getDistanceFormat(distance, entities)
Input: Distance in KM retrieved from distanceByLatLong in location.py, as well as the list of entities sent to distanceByLatLong in location.py.
Output: String containing a properly worded response from the bot. Return type is string.

geoInfo.py
Responsible for all weather and point of interest data.

get_temperature(entities, lat=None, long=None, wtype='celsius')
Input: List of entities containing locations.
Output: The temperature of all locations in the entity list in celsius. Return type is an array of floats.

get_weather(entities, lat = 0, long = 0)
Input: List of entities containing locations, a json type, or a location as a string.
Output: the weather of that location. Return type is string.

get_point_of_interest(entities, limit=5, latitude = 0, longitude = 0)
Input: List of entities containing locations, a json type, or a location as a string.
Output: the POI of that location. Return type is string.

main.py
Acts as a GUI for the user to enter questions and to view the chatbot's responses.