MIT Remote Artificial Intelligence Program

Yu Bai (Carl Bai) Instructor: Dr. Fan ZHANG

carlbai66@gmail.com

9/15/2019

In completing this homework, I referred to…

* Downloadable instruction and handouts in the cloud:  [https://pan.baidu.com /](http://www.cs.ucr.edu/~eamonn/cs170/)
* Rasa Tutorial:

https://rasa.com/docs/rasa/user-guide/rasa-tutorial/

* Open Weather Map API Documentation: https://rapidapi.com/community/api/open-weather-map?endpoint=53aa6043e4b00287471a2b66
* Building a chatbot with Rasa NLU and Rasa Core: <https://www.youtube.com/watch?v=xu6D_vLP5vY>
* Telegram Bot API:

https://core.telegram.org/bots/api

All important code is typed and designed by myself. Unimportant subroutines that are not completely original are…

• All subroutines used from **telebot** for telegram Chatbot connection

• All subroutines used from **time** for timing and records

• All subroutines used from **random** for reply auto-generation

# **Artificial Intelligence Program: telegram chatbot Write Up**

Yu Bai (Carl Bai)

**Introduction**

Chatbots continue to grow in popularity with 80% of businesses expecting to be using one by 2020. Though it may feel like the term ‘Chatbot’ has only recently entered the public lexicon, they actually have made a huge difference in human’s lives. When we retrospect the chatbots released recently, Facebook have recently launched ‘Chat Extensions’ which lets you use Messenger bots in group chats so you and your friends can interact with a bot together. Apple have finally got round to releasing there long rumored ‘HomePod’ and Amazon’s next-gen Echo, Show, now includes a screen. As the technology continues to evolve, more AI and speech based bots will enter the market. So it’s really a crunch time for us to learn more about the chatbots, including their construction and the deep knowledge.

**My Chatbot**

Based on the platform of telegram, I built up a Chatbot using the pyTelegramBotAPI. It allowed me to connect to the telegram with the given token by Telebot, a chatbot in Telegram. With the connection, I can update each message sent or received from the client. Then I can analyze and process these data to manage my own chatbot returning the corresponding results. By introducing the Rasa\_NLU for understanding user messages in natural language and extracting useful information from the text.

**Features One —— Weather**

Command weather inquiring

The command inquiring section is designed in specified commands. By choosing the domain you want, you can get the information step by step. When you enter the “#weather” the robot will ask the city you wanna inquire, then the country. With these both key words gotten, the robot will searching in the cloud, and return the current weather data to the user.

Message analysis and generation

Intent & entity recognition

Importing the Rasa, I built up the model, configuration and data. By training enough data in the data.json file, the interpreter can recognize the intent and entity from each message the user send. With these intent and entity, we can extract the most valuable key words from user’s message, and then respond with the best information the user wanna know. The message will show on the chat box on telegram.

Current weather data

If the user wanna know the current weather, they will send the message like: “Show me the weather in Dublin, please” or “Tell me the weather in Tokyo, Japan”. So the messages like these kind of format will return the intent as “inform” and the entity will be the **location.**  Some people will not tell the accurate or even no location, such as: “what’s the weather like at the moment?”, then the robot will ask: “can you tell me the location you wanna know?” so by this addition inquiry, the bot will get the location and search it on the backend. Then the best answer will appear on the chatting box.

Forecast weather data

Since somebody may wanna get the weather not only currently but also in the future, the robot is supposed to recognize whether the user wanna forecast the weather or not. Thus it’s important to have another intent as “predict” and “**time**” is a really valuable entity for analysis. In this case, commonly the message will return two important entities: “location” and “time”. With these two entities given, we can also get the information from the backend, and similarly in the next steps.

Negative message judgement

For the weather part, I was wondering in which case we need to deny the information from the robot until one day I saw a news that two people, who booked their flights online, made the simple mistake of selecting Glasgow in America instead of the UK city they wanted to fly from. Consequently, if the weather of the place is not from the country we want, then we can reject it and let it return a new information from another country.

For instance, is we ask “what’s the weather like in Glasgow?” it may inform me the weather in Glasgow, UK, then I can say “no, it’s in US” And that way, the correct answer will be given.

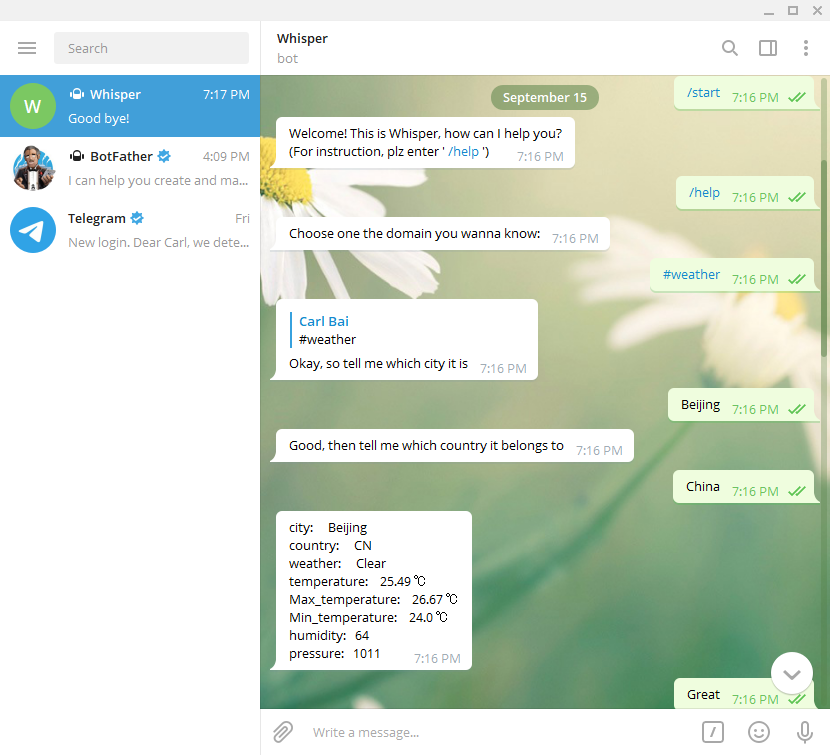
Random reply generation.

If the robot can only reply with the same words, it will be dumb and dull. So if the robot can reply with different responses, the user will feel like talking with a real man. The method of this is to have a dictionary with intent – responses. For each intent, we will have various responses, and we can randomly choose one for the reply.

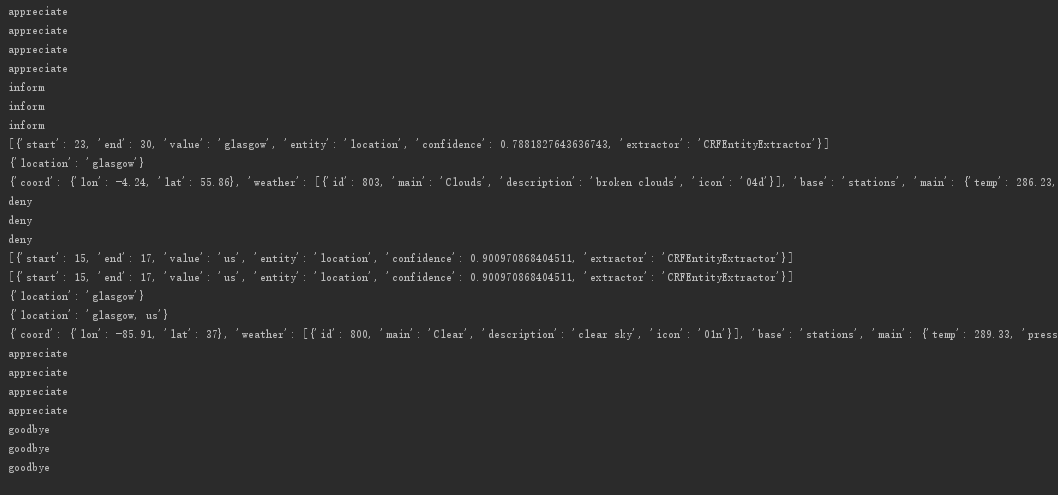
**Features Two & Three —— Movie & Game**

These two features are still engaged, the problems I need considers includes how to distinguish different functions and how to train data in different configuration. These features will be realized in the future

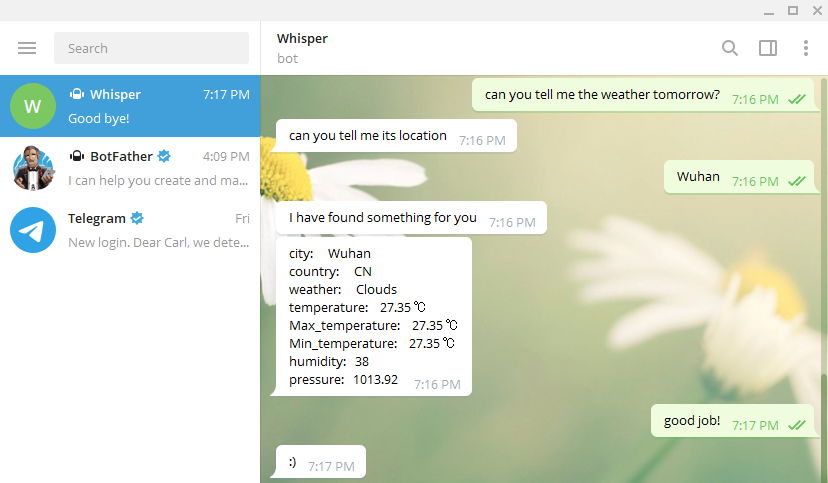
Result

**Command weather inquiring:**

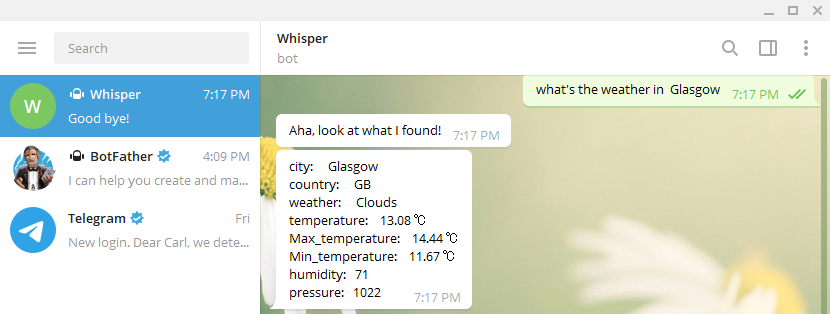
**Intent & entity recognition:**



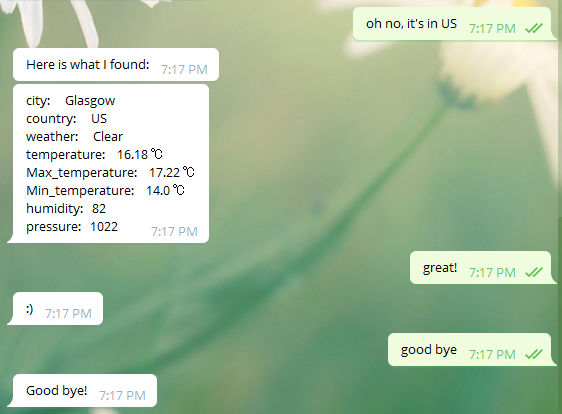
**Forecast weather data:**

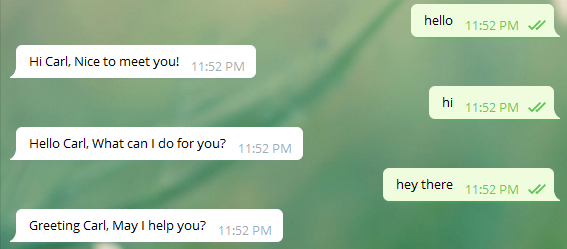


**Current weather data:**



**Negative message judgement:**

**Original Random reply generation:**



**Conclusion**

By building the Rasa-based telegram chatbot, I analyzed several data and generate the replies as shown above. Considering the list of these features: we can draw the conclusions that:

* It can be seen that from all these features, the more data we have as the training data, the more accurate intent we have as the results.
* For different kind of results, we need to figure out the entity respectively. Otherwise, we will get the wrong or inaccurate data from the cloud.
* There is no need to set the word as the decision condition, if the intent is accurate enough, with the quantities of data’s supporting, the intent is much more flexible for condition judgement.
* Since the size of training data is not enormous enough, we cannot recognize some of the message, and some message will not run correctly.
* Machine learning is a good way for AI training, but it’s not definitely the best method in small size data support.