

Final Project Report

Chatbot on Airbnb in New York City

ZEYU CHEN

Table of Contents

Introduction:	2
Background:	2
Motivation:	2
Methodology:.....	3
Languages and APIs:	3
Data collections:.....	3
Algorithms:.....	3
Regular expressions:	3
State machines:.....	5
Rasa NLU:	6
Conclusion:.....	7
Acknowledgement:	7
Future Improvements:	8
References:	9

Introduction:

Background:

Nowadays, there are lots of different kinds of chatbots near our lives, which facilitate our lives in all fields. For example, Siri from Apple, Little Ice from Microsoft, Google Assistant from Google, and Alexa from Amazon, they are all great chatbots examples. In this course, a lot of interesting algorithms and mythologies about Natural Language Processing (NLP) in Chatbot field are taught. In the final project, a simple chatbot like chatbots listed above will be implemented by using some techniques taught.

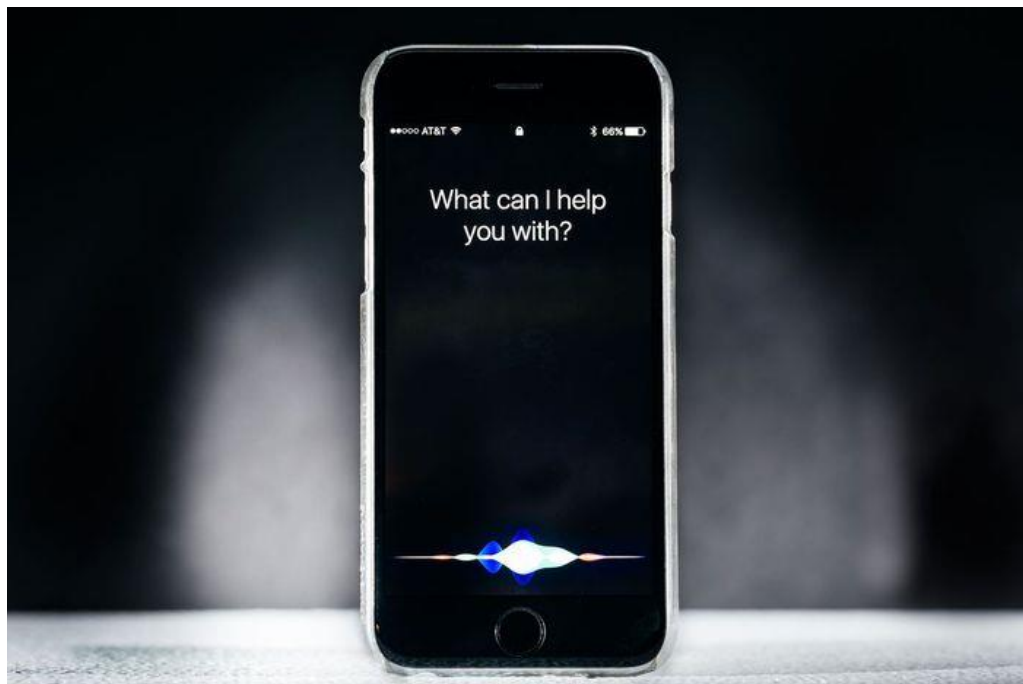


Figure 1. Siri (1)

Motivation:

With developing in Airbnb market, more and more travelers willing to choose to travel with Airbnb instead of hotels, due to its lower price and convenient locations. However, in large city like NYC, there are thousands of Airbnb, it usually costs travelers lots of time to find the Airbnb they want. The motivation of this project is creating a chatbot that can help travelers find the Airbnb they want.

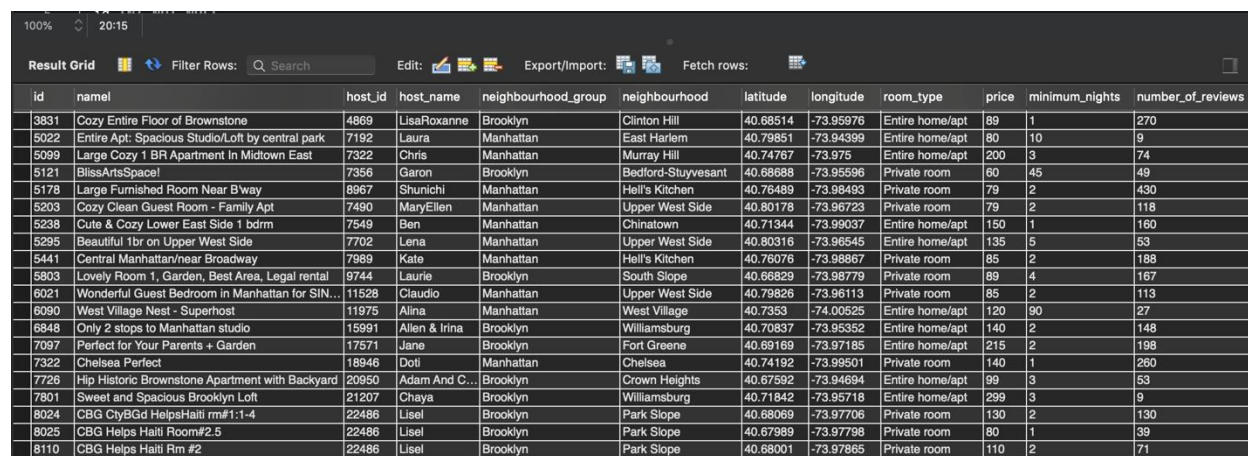
Methodology:

Languages and APIs:

In this project, the algorithms will be mainly implemented by python through PyCharm, for the data, all the data about the Airbnb will be stored in the MySQL database, and MySQL-connector will be used to interact between the python models and the backend database. The GUI of the chatbot will use the GUI of WeChat through WXPY API.

Data collections:

All the datasets about Airbnb are downloaded from Kaggle which a professional datasets provider is. To improve the data searching and matching speed, all those data were exported to a database through a small function. To increase the accuracy of the data, preprocessing like removing invalid or incomplete data and data formatting were doing on those data. The final dataset looks like in the Figure 2.



id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270
5022	Entire Apt. Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9
5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40.74767	-73.975	Entire home/apt	200	3	74
5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68688	-73.95596	Private room	60	45	49
5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40.76489	-73.98493	Private room	79	2	430
5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.80178	-73.96723	Private room	79	2	118
5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Chinatown	40.71344	-73.99037	Entire home/apt	150	1	160
5295	Beautiful 1br on Upper West Side	7702	Lena	Manhattan	Upper West Side	40.80316	-73.96545	Entire home/apt	135	5	53
5441	Central Manhattan/near Broadway	7989	Kate	Manhattan	Hell's Kitchen	40.76076	-73.98867	Private room	85	2	188
5803	Lovely Room 1, Garden, Best Area, Legal rental	9744	Laurie	Brooklyn	South Slope	40.66829	-73.98779	Private room	89	4	167
6021	Wonderful Guest Bedroom in Manhattan for SIN...	11528	Claudio	Manhattan	Upper West Side	40.79826	-73.96113	Private room	85	2	113
6090	West Village Nest - Superhost	11975	Alina	Manhattan	West Village	40.7393	-74.00525	Entire home/apt	120	90	27
6848	Only 2 stops to Manhattan studio	15991	Allen & Irina	Brooklyn	Williamsburg	40.70837	-73.95352	Entire home/apt	140	2	148
7097	Perfect for Your Parents + Garden	17571	Jane	Brooklyn	Fort Greene	40.69169	-73.97185	Entire home/apt	215	2	198
7322	Chelsea Perfect	18946	Doti	Manhattan	Chelsea	40.74192	-73.99501	Private room	140	1	260
7726	Hip Historic Brownstone Apartment with Backyard	20950	Adam And C...	Brooklyn	Crown Heights	40.67592	-73.94694	Entire home/apt	99	3	53
7801	Sweet and Spacious Brooklyn Loft	21207	Chaya	Brooklyn	Williamsburg	40.71842	-73.95718	Entire home/apt	299	3	9
8024	CBG CyBGd HelpsHaiti rm#1:1-4	22486	Lisel	Brooklyn	Park Slope	40.68069	-73.97706	Private room	130	2	130
8025	CBG Helps Haiti Room#2.5	22486	Lisel	Brooklyn	Park Slope	40.67989	-73.97798	Private room	80	1	39
8110	CBG Helps Haiti Rm #2	22486	Lisel	Brooklyn	Park Slope	40.68001	-73.97865	Private room	110	2	71

Figure 2. Screenshot of Database

Algorithms:

Regular expressions:

All chatterbots have an idea of a "Knowledgebase" or script, which tells the 'bot what keywords or phrases to recognize. The corresponding output is then sent to the user when a match is found. The knowledgebases are usually created by using a collection of rules, and the rules must have one or multiple inputs to recognize what the user says. To make the matching between inputs and rules easily, regular expressions can be used as basis.(2)

For example, let's say there is an input which is:

What is your address?

The resulting regular expression generated by the engine looks like this:

```
^(|.*?\b|.??\s)What\b.+?\bis\b.+?\byour\b.+?\baddress(|\b.*?|\s.*?)$
```

Basically, the engine has put wildcards between words, as well as at the start and end of the phrase. This regular expression will match to, "So, what the heck is your address?" but not to, "What's your address?" With the help of regex, the chatbots can recognize a larger range of languages who share the same regular language. This does save huge amount of time compare to hard code all the possible expressions.

```
rules = {'I want (.*)': ['What would it mean if you got {0}',  
                        'Why do you want {0}',  
                        "What's stopping you from getting {0}"],  
        'do you remember (.*)': ['Did you think I would forget {0}',  
                                  "Why haven't you been able to forget {0}",  
                                  'What about {0}',  
                                  'Yes .. and?'],  
        'do you think (.*)': ['if {0}? Absolutely.',  
                              'No chance'],  
        'if (.*)': ["Do you really think it's likely that {0}",  
                   'Do you wish that {0}',  
                   'What do you think about {0}',  
                   'Really--if {0}']  
}
```

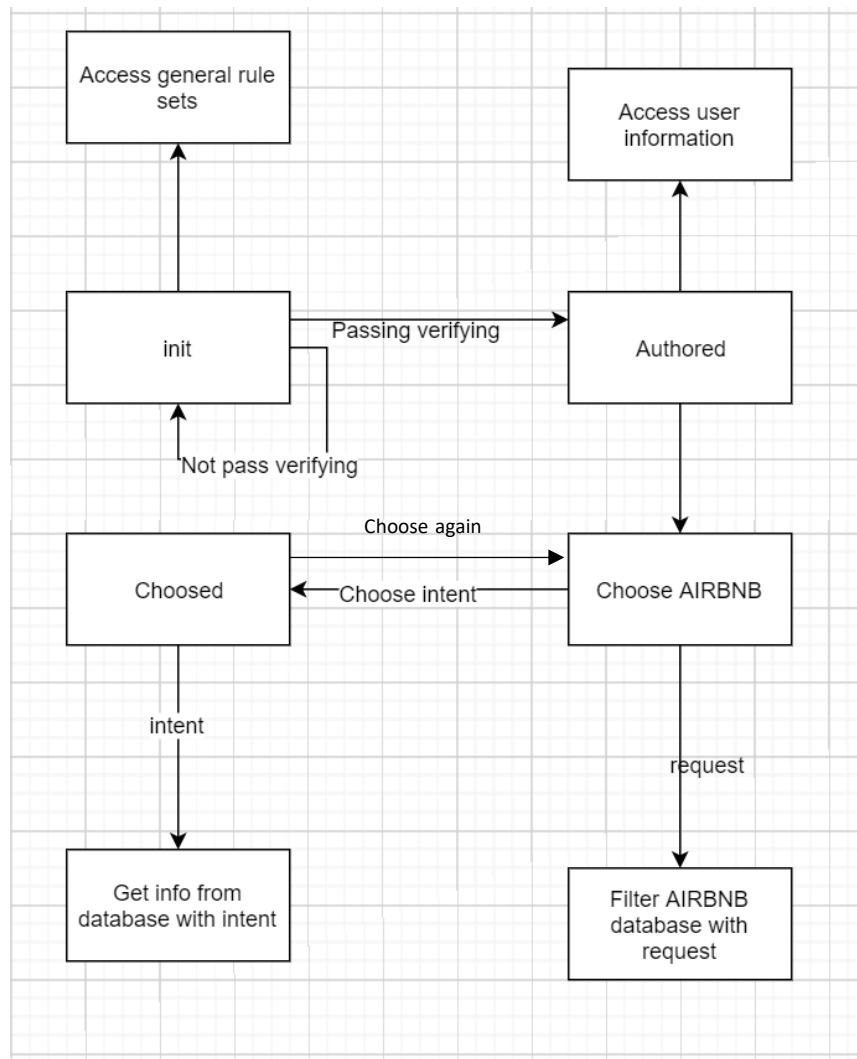
Figure 3. regex example

Another example of using regex in knowledgebase is shown above in the Figure 3. To make the Chatbots more like a human, a rule can have multiple outputs and every time the output should be picked randomly. Additionally, to avoid some questions that asked by the users not in the knowledgebase, a default general answer can be set.

To make the main code clean and more readable, a .xml file was created to save all the knowledgebases.

State machines:

To make sure the chatbots perform different functionality at different occasions, different rules might be used. To distinct different occasions, a state machine was used, with help of state machine, the chatbot can also perform a simple test and verify processing. The following graph show the states diagram of our Chatbot.



In general, there are 4 states: init, authored, Choose AIRBNB, and Chooosed. These state transitions will be toggled by specific intents of the users. And in each state, the user will have different level of information access.

Init: In this state, the robot will only answer some general questions which were listed in the general rules sets. If the user mention word “Airbnb”, the state transition between **Init** and **Authored** will be triggered. The Chatbot will ask the user for their

phone number, the verification will pass if the phone number is valid and the state will switch to **Authored** as a result. However, if the number is illegal, the state will not change.

Authored: In this state, the chatbot can access some information left by the user, if the user is an old user. Meanwhile, the chatbot will ask the user about some information about the Airbnb. The state transitions will be triggered when the chatbot query the database by using the information provided by the user.

Choose AIRBNB: As the chatbot will give the user some suggestions based on the information provided by the users; this state will be used for the user to choose the Airbnb. Once a specific Airbnb is chosen, the chatbot will query the database again to get the detailed information about that Airbnb. At the same time, the state will be changed to **Chooosed**.

Chooosed: This is the regular end state, however, if the user is not satisfied with the Airbnb he/she chosen before, the state will go back to the **Choose AIRBNB** state again. If the user satisfies with his/her choice, the chatbot will record his/her choice as personal information.

Rasa NLU:

Rasa NLU is an open-source natural language processing tool for intent classification, response retrieval and entity extraction in chatbots. For example, taking a sentence like (3):

```
"I am looking for a Mexican restaurant in the center of town"
```

and returning structured data like

```
{
  "intent": "search_restaurant",
  "entities": {
    "cuisine": "Mexican",
    "location": "center"
  }
}
```

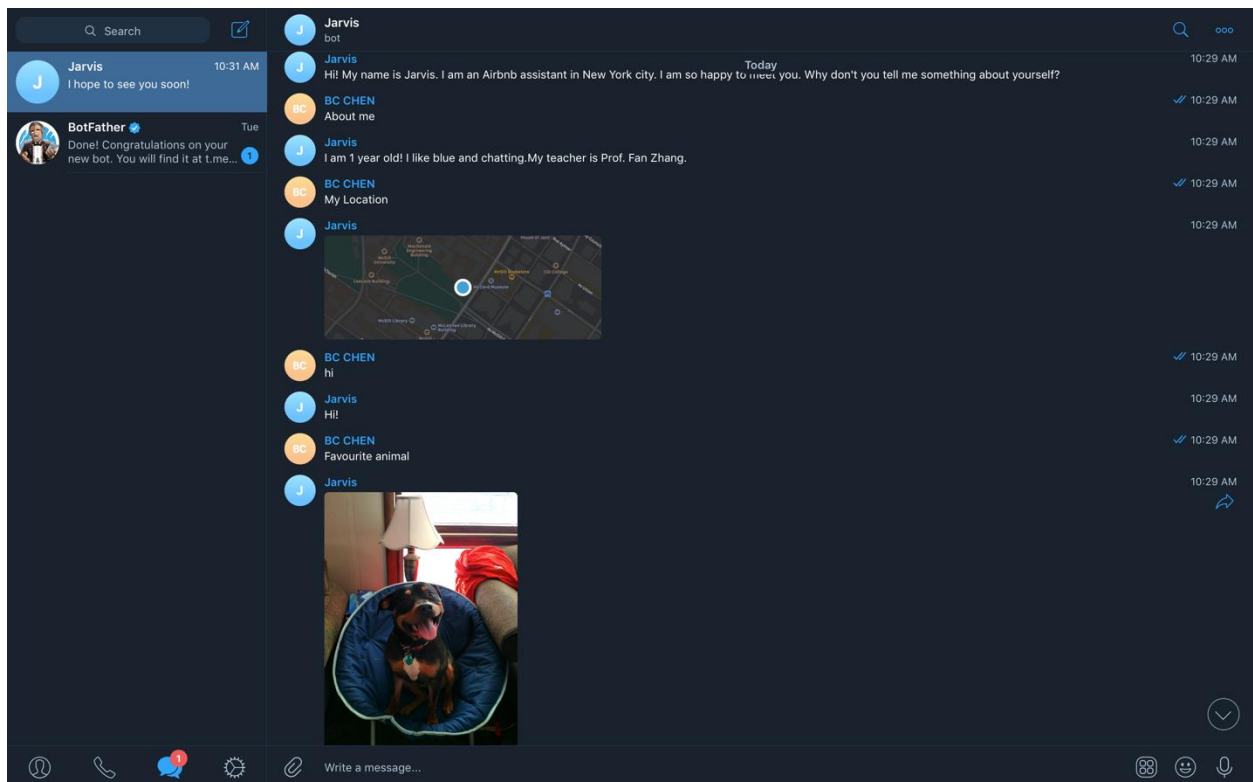
Figure 4. Example from Rasa NLU doc

This package helps analyze the intent of a message, which contributes to classify the intents to some general kinds. This can reduce the total number that Chatbot needs to

recognize and decrease the difficulty of Implementation of the Chatbot. Additionally, it helps the chatbot to recognize the triggers between states.

Conclusion:

In conclusion, my robot does a quite well job on Airbnb assisting and searching. The GUI of my project is like following:



During this project, I have learned a lot of interesting algorithms. I improved my debugging ability and the ability of using different APIs and packages.

Acknowledgement:

Thanks Prof. Fan Zhang for advising and helping in this project.

Future Improvements:

Currently, all the data provided by the Chatbot are getting from the database which might be not the latest. Next generation chatbot may be linked to online source which will be updated frequently.

Responding speed and program efficiency is another aspect that can be improved.

References:

1. <https://www.google.com/imgres?imgurl=https%3A%2F%2Fasset-a.grid.id%2Fcrop%2F0x0%3A0x0%2F700x465%2Fphoto%2F2019%2F06%2F12%2F2021461172.jpg&imgrefurl=https%3A%2F%2Finfokomputer.grid.id%2Fread%2F121752505%2Fterungkap-mengapa-otak-apple-siri-lebih-dungu-dibanding-amazon-alexa&docid=bNkToY-ToR1Q9M&tbnid=CuNn3tlgvXwIIM%3A&vet=1&w=700&h=465&bih=1171&biw=1163&ved=2ahUKEwjPzZm14oPmAhUlm-AKHV-OChgQxiAoC3oECAEQLQ&iact=c&ictx=1>
2. <https://www.codeproject.com/Articles/18109/Building-an-AI-Chatbot-using-a-Regular-Expression>
3. <https://rasa.com/docs/rasa/nlu/about/#>