Chat bot Project

Zuquan Chen
597359489@qq.com

***Abstract*—This report aims to briefly analyze the working principle of natural language processing chat bot and introduce an implementation of a chat bot helping people to search stock information.**

Keywords—artificial intelligence, natural language analysis, Rasa-NLU, machine learning, chat bot

I.INTRODUCTION

In today's world, Artificial Intelligence has been applied in an increasing number of fields. And natural language processing is one of the pop topics which aims to train computers with specific data and enables it to understand the intent of human daily language . There are several challenges in natural language processing including entities extracting, natural language understanding, and natural language generation. This report focuses on the application of natural language understanding and gives some examples on how to implement it in a chat bot which helps to search the stock information including the market capitalization, real-time price and volume of transactions in a given day with the help of Python.

II. WORKING PRINCIPLE

(1) Echo Respond

Echo-bot is a bot without intelligence, it can only extracts the content of user's input and responds the input using this content in a already made format. In other words, the response of the Echo-bot is totally depending on the user's input and lack personality.

Example:

User: Hello!

Bot: I can hear you! You said Hello

However the Echo-bot can be made more personalized.The solution is defining several answers for a question and using the random function to select one of the answers randomly. In this case, the answer is more flexible -the same question may generate different responses and human-like.

Example1:

User: What's your name !

Bot: My name is Echobot!

Example2:

User: What's your name !

Bot: They call me Echobot!

We can also use the regular expression represents to improve our Echobot. The method is that we extract specific content when use's input match the regular expression and then transform sentences grammatically

Example:

User: I want a hamburger

Bot:Why do you want a hamburger

(2) Intent Recognition and Entity Extraction

It is important to understand the intent and extract entities in natural language. This part of work can also be achieved by regular expression. It is simpler and has higher efficiency than a machine learning approach. However, it is difficult to define a good regular expression. This report mainly focuses on the machine learning approach. The machine will understand users' intents by learning from the training data. To make the data readable to machine, it is essential to transform natural language into word vectors.

- Intent recognition

Supervised learning is used to recognize the intent in natural language. The training data is a set of word vectors with label intents. There are kinds of the algorithm can be used to train the model. For example, KNN (K nearest neighbour) and SVM (Support-Vector machine). The work efficiency depends on the size of the training data set.

- Entity Extraction

For extracting entities in natural language, the keyword is not suitable for extracting unknown entities. In other words, it is impossible to define every entity previously. Others method such as spelling, context and after specific words need to be used. In addition, It is important to identify the roles entities are playing. For example:

I want a flight from Tei Aviv to Bucharest.

Show me flights to Shanghai from Singapore.

In this example, Tei Aviv, Bucharest, Shanghai and Singapore are all location entities while the role they played is different: place of departure and destination. Dependency analysis is another factor need to be considered. When considering the role of the entity plays, words related to it also need to be considered. More specific examples will be shown in the implementation of part.

III. IMPLEMENTATION

(1) Define State Rules

The state of the chat bot is plays a key role in achieving the project. In view of the function isn't very complex only three states are defined. There are INIT, CHOOSE_FUNC and CHOOSE_TIME.

Choose company is an equally important aspect, but I use other methods to achieve it rather than using state_machine

The chatbot does not do any work about stock information search when the state is INIT. The state CHOOSE_FUNCTION means users have selected a specific function provided by the chatbot and get the result. When the time is needed (for example get volume of transaction in a given day) and use's haven't given valid time(including giving wrong time or forgetting offering a time),the state

will turn to CHOOSE_TIME

The code is as shown below:

```
# Define the INIT state
INIT=0

# Define the CHOOSE_FUNC state
CHOOSE_FUNC=1

# Define the CHOOSE_ state
CHOOSE_TIME=2

# Define the policy rules
policy = {
    (INIT, "ask_for_information"): (CHOOSE_FUNC, "we can offer you information including real-time price, volume of transaction and the market capitaliz
    (INIT, "none"): (INIT, "I'm sorry - I'm not sure how to help you, I am a chat_bot who can provide you with some information about an stock, please a
    (INIT, "intent_with_time"): (INIT, ""),
    (INIT, "intent_without_time"): (CHOOSE_TIME, "please tell us the detailed time, just like 2019-01-01"),
    (CHOOSE_FUNC, "intent_without_time"): (CHOOSE_TIME, "please tell us the valid and detailed time, just like 2019-01-01"),
    (CHOOSE_FUNC, "intent_with_time"): (INIT, ""),
    (CHOOSE_FUNC, "none"): (CHOOSE_FUNC, "I'm sorry - I'm not sure how to help you?"),
    (CHOOSE_TIME, "intent_without_time"): (CHOOSE_TIME, "please tell us the valid and detailed time, just like 2019-01-01"),
    (CHOOSE_TIME, "intent_with_time"): (INIT, ""),
}
```

This module is to define how the state change.

```
def send_message(policy, state, intent, time):
    #print("USER : {}".format(message))
    #print(state, intent)
    new_state, response = respond(policy, state, intent, time)
    #print("BOT : {}".format(response))
    return new_state, response

def respond(policy, state, intent, time):
    (new_state, response) = policy[(state, interpret(intent, time))]
    return new_state, response

intents = {"real_time_price", "market_capitalization"}
def interpret(intent, time):
    if intent == "ask_for_information":
        return intent
    if intent in intents:
        return "intent_with_time"
    if intent != "volume_of_transactions":
        return 'none'
    if len(time) == 0:
        return "intent_without_time"
    else:
        return "intent_with_time"
```

(2) Define Training Data and Model

The approach to achieve extracting intent and entities is machine learning. Rasa-NLU ,however in order to use more technology I have commended I use Rasa-NLU only for extracting intent and use Spacy to extract entity . First,we should install Rasa-NLU packages in the environment and import it in the project. Rasa-NLU accepts different kinds of training data. The data type used in this project is Json which is more readable and has a clear structure. The standard
format of JSON type data is shown below:

```
{
  "text": "i'm want to know something about AAPL to help me to analyse it ",
  "intent": "ask_for_information",
  "entities": []
},
{
  "text": "show me some details about TSLA",
  "intent": "ask_for_information",
  "entities": []
},
```

The next step is using training data to generate an interpreter

```
from rasa_nlu.training_data import load_data
from rasa_nlu.model import Trainer
from rasa_nlu import config

training_data = load_data('training_data\demo-rasa.json')
trainer = Trainer(config.load('config\config_spacy.yml'))
trainer.train(training_data)
```

The final step is to call Rasa-NLU and get the user's intent

```
result = interpreter.parse(message)["intent"]
```

(3) Extract entities

After obtain the use's intent by Rasa-NLU, what we need is to extract the entities in use's message so we can know how to call API and return expected information. In order to show as many as techniques as possible, I choose Spacy rather than Rasa-NLU to extract entities

The code is shown below

```
import spacy
nlp = spacy.load('en_core_web_md')
include_entities = ['ORG']

# Define extract_entities()
def extract_entities(message):
    # Create a dict to hold the entities
    ents = dict.fromkeys(include_entities)
    # Create a spacy document
    doc = nlp(message)
    for ent in doc.ents:
        if ent.label_ in include_entities:
            return ent.text
    return ""
```

(4) Extract time

Calling the get_historical_data(), we still need detailed and valid date, In order to get date in specific format, I choose regular expression to achieve this function. When obtained the date, I also compare it with present date to make sure the date we extract is valid.

The code is shown below.

```
import re
from datetime import datetime

def checkValidTime(time):
    format_pattern = "%Y-%m-%d"
    end_date = datetime.now()
    end_date = end_date.strftime(format_pattern)
    diff = datetime.strptime(end_date, format_pattern) - datetime.strptime(time, format_pattern)
    if diff.days > 0:
        return True
    else:
        return False

def getTime(message):
    pattern = re.compile(r'\b\d{4}-\d{1,2}-\d{1,2}\b')
    results = pattern.findall(message)
    if results:
        #print(results[0])
        if checkValidTime(results[0]):
            return results[0]
        else:
            return ""
    else:
        return ""
```

(5)Connect to iexfinance

The chatbot created in this project aims to help the user get useful information of specific stock. To achieve this function we are supposed to import iexfinance API. Like installing Rasa-NLU,we need to install iexfinance to the Python environment and import it to the project.

```
1  from iexfinance.stocks import Stock
2  from iexfinance.stocks import get_historical_data
```

With the help of iexfinance we can design several functions to call the API and get expected data from it. Let us suppose that we have obtained the entity(the name of target company) and time(if needed) from other modules

```
def getVolume(entity, time):
    f = get_historical_data(entity, token='sk_f384a347e6774c6e96755b56e298b8ad')
    print(f.loc[time]['volume'])
    return f.loc[time]['volume']

def getValue(entity, intent, time):
        ent = Stock(entity, token='sk_f384a347e6774c6e96755b56e298b8ad')
        if intent == "real_time_price":
            return "the real time price of the target company is {0}".format(ent.get_price())
        elif intent == "volume_of_transactions":
            print("getValue", time)
            return "the volume of transaction in that day of target ORG is {0}".format(getVolume(entity, time))
        elif intent == "market_capitalization":
            return "the total market_capitalization of target ORG is {0}".format(ent.get_market_cap())
        else:
            return "sorry, i don't know,maybe there are some problems with iexfinance, please choose another questions?"


def getVolume(entity, time):
    f = get_historical_data(entity, token='sk_f384a347e6774c6e96755b56e298b8ad')
    print(f.loc[time]['volume'])
    return f.loc[time]['volume']
```

(1) We can get real-time price by calling ent.get_price()

(2) We can get the volume of transactions by calling get_historical_data, which will return some extra data, so we use f.loc[time][‘volume’] to restrict the date and kinds of data.

(3) We can get the market capitalization by calling get_market_cap()

Connect to WeChat

The final module is about connecting the program to WeChat , the most well-known online chat application in China. The tools used in this project is WXPY which is a powerful tool to process messages on WeChat. As usual,the first step is to install the packages of WXPY to the Python environment and import it to the project.

```
from wxpy import *
```

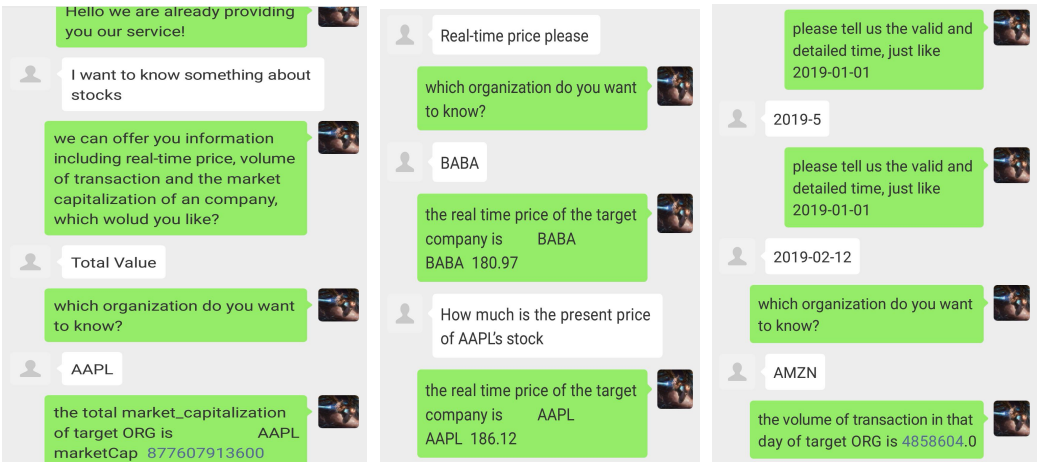
First, initialize a **Bot()** which can provide a QR code for the user to scan and then log in their WeChat account. In order to keep receiving the message from a specific friend or a group of them, registering the bot is a good choice. Then define a method called **reply_my_friend(msg)** to accept messages and send the answers to users. The codes are shown below:

```
bot = Bot()
my_friend = bot.friends().search('陈祖泉', sex=MALE)[0]
my_friend.send('Hello we are already providing you our service!')
@bot.register()
def print_others(msg):
    print(msg)
```

In order to find your own friends . You are supposed to replace the parameter in the bot.friends().search(‘ nick name of your friend’,sex=)

IV. EVALUATION

The result is shown below



With using according API, The stock chatbot can choose different responses denpending on the questions uses asked. When it comes to some irrelevant topic, the chatbot can also inform users. However, there are still some limitations for this simple chatbot, such as the response is still a little bit stiff or sometimes the intent extracted by it is not too accurate due to the relatively less training data.

V. CONCLUSION

In conclusion, this report explains some of the working principles about how to create an usual chatbot and gives the process of finishing a simple stock chatbot. The result shows that good chatbot can help users efficiently get useful information. It can be applied in various area and have great potential.

Although this chatbot is relatively simple compared with some well-known AI like siri, the process of building it is not that easy .I have learned a lot about and my programming power is enhanced in this process.

I am firmly convinced that language analysis chatbot have a promising future and can be equipped with real intelligence in one day.