MIT Remote Research Report
(Stock inquiry robot)

1.the project background

What is artificial intelligence, the official definition is: artificial intelligence is a new technical science that studies and develops theories, methods, techniques, and applications for simulating, extending, and extending human intelligence.

Artificial intelligence is a branch of computer science that attempts to understand the essence of intelligence and produce a new intelligent machine that responds in a manner similar to human intelligence. Research in this area includes robotics, speech recognition, image recognition, Natural language processing and expert systems. Since the birth of artificial intelligence, the theory and technology have become more and more mature, and the application field has been expanding. It is conceivable that the technological products brought by artificial intelligence in the future will be the "container" of human wisdom. Artificial intelligence can simulate the information process of human consciousness and thinking. Artificial intelligence is not human intelligence, but it can be like human thinking, and it may exceed human intelligence.

Artificial intelligence is a challenging science, and people who do this work must understand computer knowledge, psychology, and philosophy. Artificial intelligence is a very broad science that consists of different fields, such as machine learning, computer vision, etc. In general, one of the main goals of artificial intelligence research is to make machines capable of doing what is usually required by human intelligence.

Artificial intelligence has been known as one of the world's three cutting-edge technologies (space technology, energy technology, artificial intelligence) since the 1970s. It is also considered to be one of the three cutting-edge technologies (genetic engineering, nano science, artificial intelligence) in the 21st century. This is because it has developed rapidly in the past three decades, has been widely used in many subject areas, and has achieved fruitful results. Artificial intelligence has gradually become an independent branch, both in theory and in practice. Into a system.

Therefore, in my mind, artificial intelligence will be widely used in the future of human life. Moreover, this technology will certainly become an important pillar of society, greatly reducing many of the repeated and inefficient work of human beings, making the whole society more efficient. Therefore, I feel that it is very necessary to learn programming in artificial intelligence.

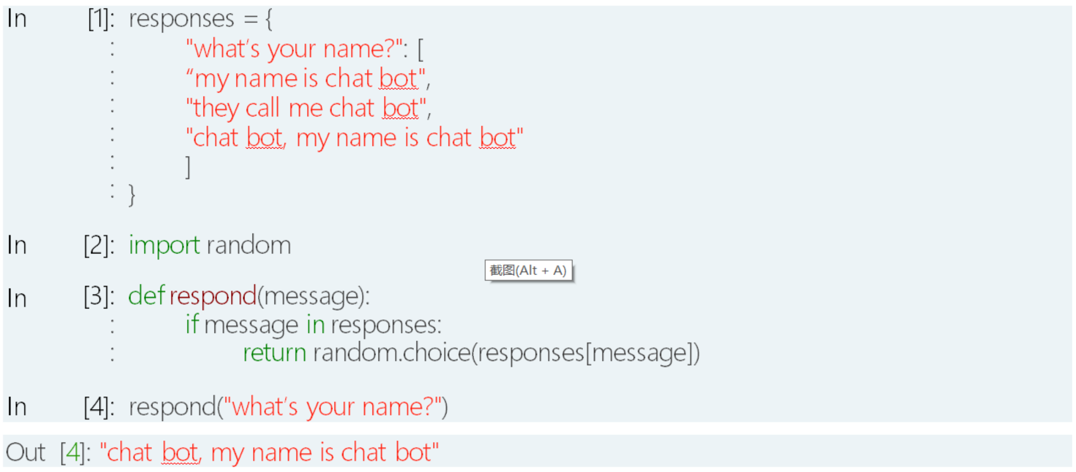
Although my undergraduate major is automotive engineering, I don't want to do research in the field of traditional machinery. I hope that I can study some interdisciplinary fields, such as the combination of computer and machinery. At this time, I thought of automatic driving, and then I thought of artificial intelligence. However, my did not learn the knowledge of artificial intelligence during the bachelor, but only learned the C++ programming language and computer basics, so I decided to learn from Mr. Fan's knowledge and programming methods.

After a month of study, Mr. Fan gave us a detailed explanation of how to use Python, and also explained in depth how to make a chat bot. Mr. Fan let us understand the development history of Python, the development and role of artificial intelligence. Mr. Fan also showed us how to program, taught us how to write the simplest robot that can automatically reply to information, how to call functions, how to define functions yourself, and so on. After finishing this month's class, I gained a lot of knowledge that I couldn't learn at school, and I also met many new friends.

2. the technology which included in the project

In this month's class, Mr. Fan taught us a lot of things, when summed up, there are probably the following points. (The pictures below are examples)

(1) Multiple selective answers to the same question and provide a default answer.



```
In      [1]:  responses = {
            :        "what's your name?": [
            :        "my name is chat bot",
            :        "they call me chat bot",
            :        "chat bot, my name is chat bot"
            :        ]
            : }

In      [2]:  import random

In      [3]:  def respond(message):
            :        if message in responses:
            :              return random.choice(responses[message])

In      [4]:  respond("what's your name?")
Out  [4]: "chat bot, my name is chat bot"
```

This is a statement that gives the robot the ability to answer questions randomly. Simply put, several different answers to a question are written in advance, and then the robot can choose from these answers when answering the question. And this choice is random, so it looks natural and intelligent.

(2) Can answer questions through regular expressions, pattern matching, keyword extraction, syntax conversion, etc.

Regular expressions, also known as regular expressions. Regular expressions are often abbreviated as regex, regexp, or RE in computer programming code, a concept in computer science. Regular expressions are often used to retrieve and replace text that conforms to a pattern (rule).

Many programming languages support string manipulation using regular expressions. For example, a powerful regular expression engine is built into Perl. The concept of regular expressions was originally popularized by tools in Unix (such as sed and grep). Regular expressions are usually abbreviated as "regex", singular with regexp, regex, plural with regexps, regexes, regexen.

In simple terms, the regular expression method can be divided into three steps:

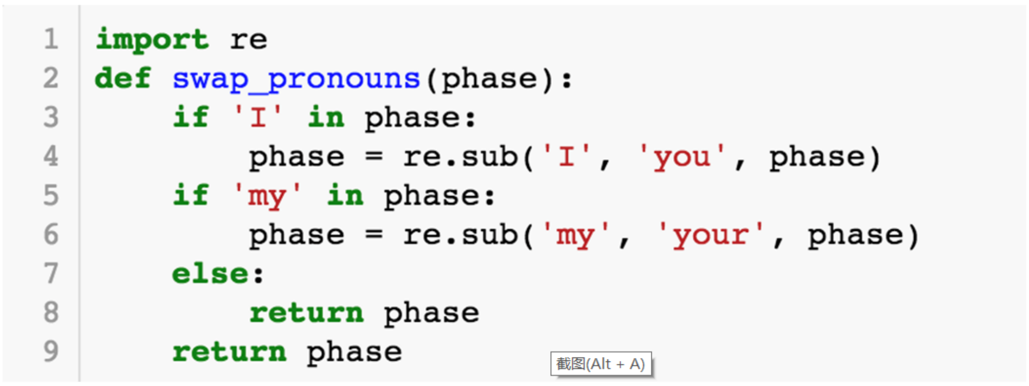
1) Determine if the given string conforms to a specific form

```
In  [1]:   import re
In  [2]:   pattern = "do you remember        .*"
In  [3]:   message = "do you remember        when you last watched the movie?"
In  [4]:   match = re.search(pattern,           message)
In  [5]:   if match:
             :         print("string matches!")
             Out[5]: string matches!
```

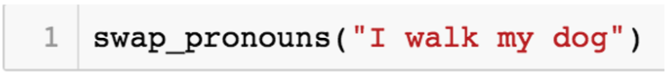
2) Extract the key characters in the string, that is, the characters that match the regular expression

```
In        [1]:  import re
In        [2]:  pattern = "if (.*)"
In        [3]:  message = "what would happen if the German team lost in the World Cup?"
In        [4]:  match = re.search(pattern, message)
In        [5]:  match.group(0)
Out [5]:        'if the German team lost in the World Cup?'
In [6]:         match.group(1)
Out[6]:         'the German team lost in the World Cup'
```

3) Convert a specific string to another string



```python
import re
def swap_pronouns(phase):
    if 'I' in phase:
        phase = re.sub('I', 'you', phase)
    if 'my' in phase:
        phase = re.sub('my', 'your', phase)
    else:
        return phase
    return phase
```



```python
swap_pronouns("I walk my dog")
```

(3) can extract user intent through one or more schemes of regular expression,

nearest neighbor classification or support vector machine.

This is an advanced method that extracts the intent of the entity and the speaker in the sentence through some of the keywords in the sentence, so that more code can be used to identify more things.

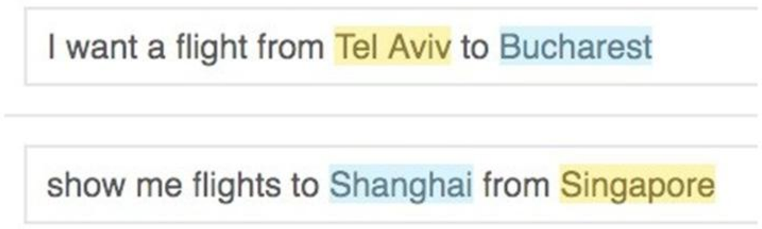
```
In [1]: pattern = re.compile('[A-Z]{1}[a-z]*')
In [2]: message = """
            Mary is a friend of mine,
            she studied at Oxford and
            now works at Google"""
In [3]: pattern.findall(message)
Out[3]: [‘Mary’, ‘Oxford’, ‘Google’]
```

(4) Named entity identification through pre-built named entity types, role relationships, dependency analysis, etc.

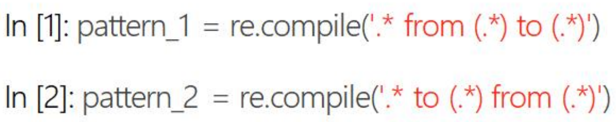
1) Identification of pre-built named entities

```
In [1]: import spacy
In [2]: nlp = spacy.load('en')
In [3]: doc = nlp("my friend Mary has worked at Google since 2009")
In [4]: for ent in doc.ents:
   ...:     print(ent.text, ent.label_)
   ...:

Mary PERSON
Google ORG
2009 DATE
```

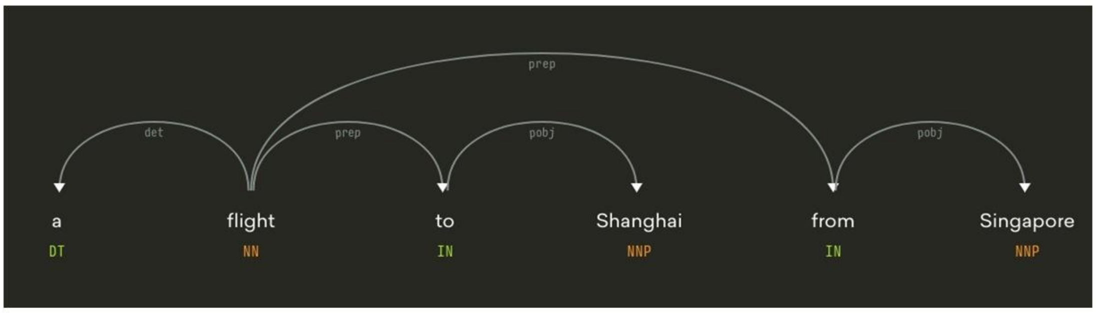
2) Role relationship

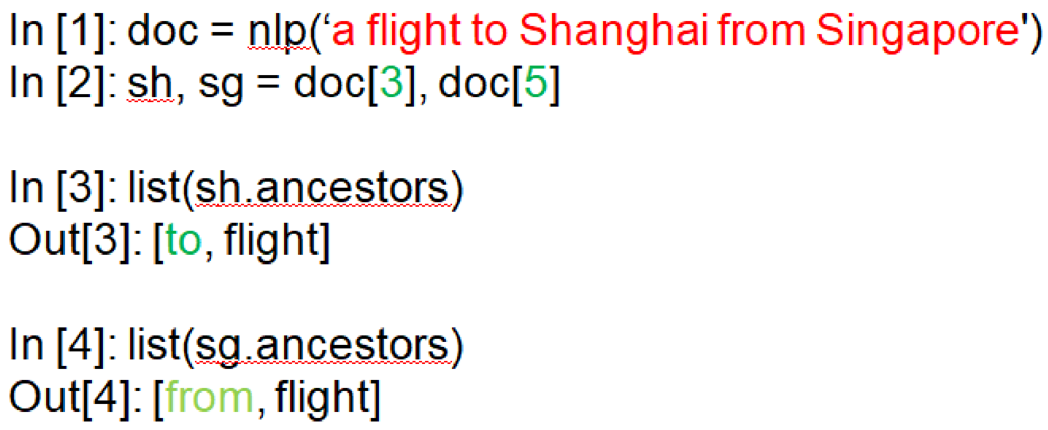

I want a flight from Tel Aviv to Bucharest

show me flights to Shanghai from Singapore


```
In [1]: pattern_1 = re.compile('.* from (.*) to (.*)')

In [2]: pattern_2 = re.compile('.* to (.*) from (.*)')
```

3.)Dependency analysis




```
In [1]: doc = nlp('a flight to Shanghai from Singapore')
In [2]: sh, sg = doc[3], doc[5]

In [3]: list(sh.ancestors)
Out[3]: [to, flight]

In [4]: list(sg.ancestors)
Out[4]: [from, flight]
```

(5) Construction of a local basic chat robot system based on Rasa NLU.

The Natural Language Understanding (NLU) system is the cornerstone of more advanced applications such as question answering systems and chat bots. The basic NLU tools include two tasks: entity identification and intent recognition.

The existing NLU tools, mostly in the form of services, solve the above two tasks by calling the remote http restful API to parse the target statement. Such tools include Google's API.ai, Microsoft's Luis.ai, Facebook's Wit.ai, and more. Kitt.ai, which was just acquired by Baidu, in addition to Baidu's voice wake-up, actually has a large part of the work on the NLU. Their inspiring Chatflow includes a NLU engine of its own.

These tools provide users with NLU services while continuously training and improving their models through a large number of labeled and non-labeled data uploaded by users themselves. For data-sensitive users, open source NLU tools like Rasa.ai open up another path for us. More importantly, such open source tools can be deployed locally, training and adapting models to actual needs. It is said that the demand for specific areas is much better than the common online NLU services.

Rasa NLU's task of entity recognition and intent recognition requires a trained MITIE model.

Rasa NLU is very suitable for the dialogue of robots used for artificial intelligence, just need to give the training model.

1)The method of use is as follows:

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


# Create a trainer
trainer = Trainer(config.load("/PATH/TO/CONFIG_FILE"))

# Load the training data
training_data = load_data('/PATH/TO/TRAINING_DATA')

# Create an interpreter by training the model
interpreter = trainer.train(training_data)
```

2)Interpreter：

```
In [1]: message = "I want to book a flight to London"
In [2]: interpreter.parse(message))
Out[2]: {
  "intent": {
    "name": "flight_search",
    "confidence": 0.9
  },
  "entities": [
    {
      "entity": "location",
      "value": "London",
      "start": 27,
      "end": 33
    }
  ]
}
```

3)Use rasa nlu to train the format of the data:

```
In [1]: from rasa_nlu.training_data import load_data
In [2]: training_data = load_data(”/PATH/TO/training_data.json")
In [3]: import json
In [4]: print(json.dumps(training_data.entity_examples[0].data, indent=2))
Out[4]: {
  "intent": "restaurant_search",
  "entities": [
    {
      "start": 31,
      "end": 36,
      "value": "north",
      "entity": "location"
    }
  ]
}
```

(6) Database query and use natural language to explore database content (extract parameters, create queries, responses)

1)Extract parameters:

```
In [1]: message = "a cheap hotel in the north"
In [2]: data = interpreter.parse(message)
In [3]: data
Out[3]:
{
        'entities’: [
                        {'end': '7’, 'entity': 'price’, 'start': 2, 'value': 'lo’},
                        {'end': 26, 'entity': 'location', 'start': 21, 'value': 'north’}
                ],
        'intent': {'confidence': 0.9, 'name': 'hotel_search’}
}

In [4]: params = {}
In [5]: for ent in data["entities"]:
...:        params[ent["entity"]] = ent["value"]

In [6]: params
Out[6]: {'location': 'north', 'price': 'lo'}
```

2)Query from the parameters:

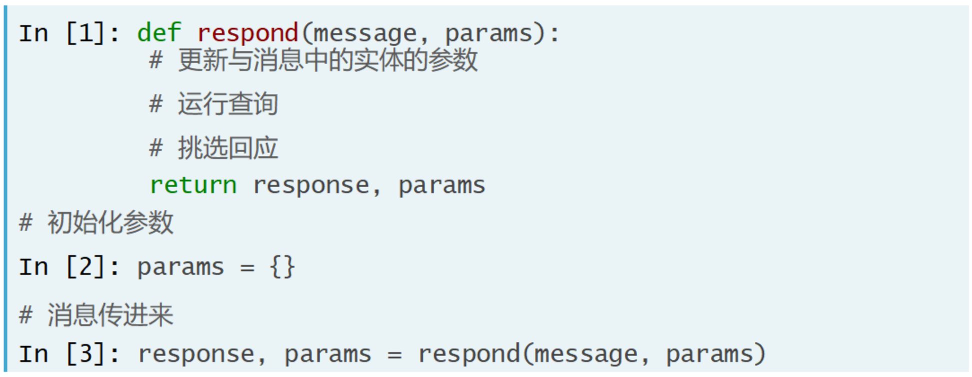
```
In [7]: query = “select * FROM hotels"
In [8]: filters = ["{}=?".format(k) for k in params.keys()]
In [9]: filters
Out[9]: ['price=?', 'location=?’]

In [10]: conditions = " and ".join(filters)
In [11]: conditions
Out[11]: 'price=? and location=?’

In [12]: final_q = " WHERE ".join([query, conditions])
In [13]: final_q
Out[13]: ‘SELECT * FROM hotels WHERE price=? and location=?'
```

(7) Single-round multiple incremental query technology based on incremental filter and screening negative entity technology.

1) Single round multiple queries:

When we query the robot for something, the robot does not give us the answer we want, so we change the conditions of the query, or add conditions, so that the robot can query again on the previous basis, repeating this process multiple times to get The result we want.

Use the params parameter to remember and keep the current query status:



```
In [1]: def respond(message, params):
            # 更新与消息中的实体的参数

            # 运行查询

            # 挑选回应
            return response, params
# 初始化参数
In [2]: params = {}
# 消息传进来
In [3]: response, params = respond(message, params)
```

2) Identify entities with negative meanings:

When we query the robot for something, the robot gives several possible solutions, some of which we don't want. When we reply to the robot with a negative statement, it can identify it and change the query condition to query. To get the results we want.

The specific operation method is that if the robot recognizes that the statement we issued contains "n't" or "not", it is determined whether the sentence is negative.

```
doc = nlp('not sushi, maybe pizza?')

indices = [1, 4]
ents, negated_ents = [], []

start = 0
for i in indices:
        phrase = "{}".format(doc[start:i])
        if "not" in phrase or "n't" in phrase:
                negated_ents.append(doc[i])
        else:
                ents.append(doc[i])
                start = i
```

(8) Implement multiple rounds of multi-query techniques for state machines and provide explanations and answers based on contextual issues.

A state machine is a directed graph consisting of a set of nodes and a set of corresponding transfer functions. The state machine runs by responding to a series of events. Each event is within the control of the transfer function belonging to the current node, where the scope of the function is a subset of the nodes. The function returns the next (perhaps the same) node. At least one of these nodes must be final. When the final state is reached, the state machine stops.

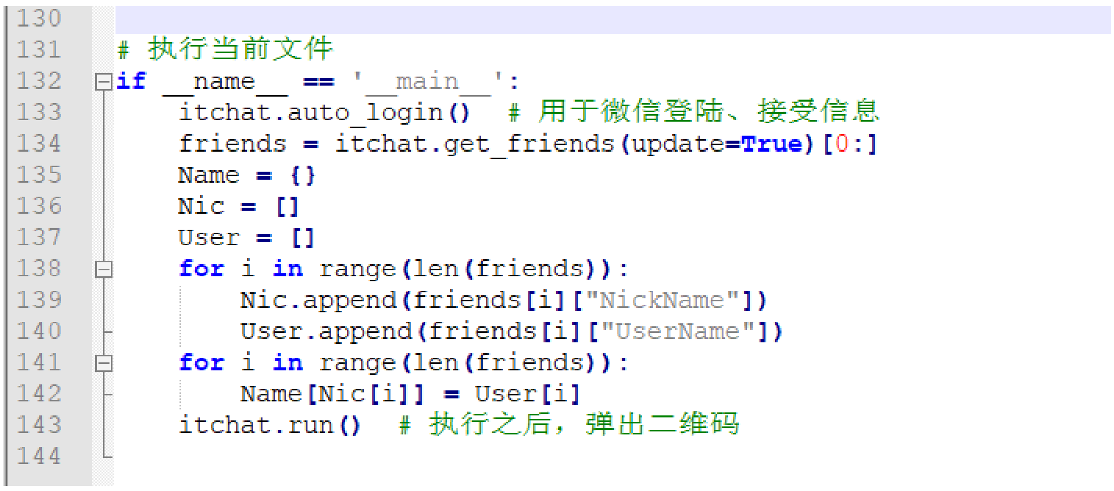
```
In [1]: state = INIT

In [2]: def respond(state, message):
                (new_state, response) = policy_rules[(state, interpret(message))] return new_state, response


In [3]: def send_message(state, message):
   ...:          new_state, response = respond(state, message)
   ...:          return new_state

In [4]: state = send_message(state, message)
```

(9) Multiple rounds of multiple query techniques for handling rejections, waiting for state transitions, and pending actions.

3.the program I wrote

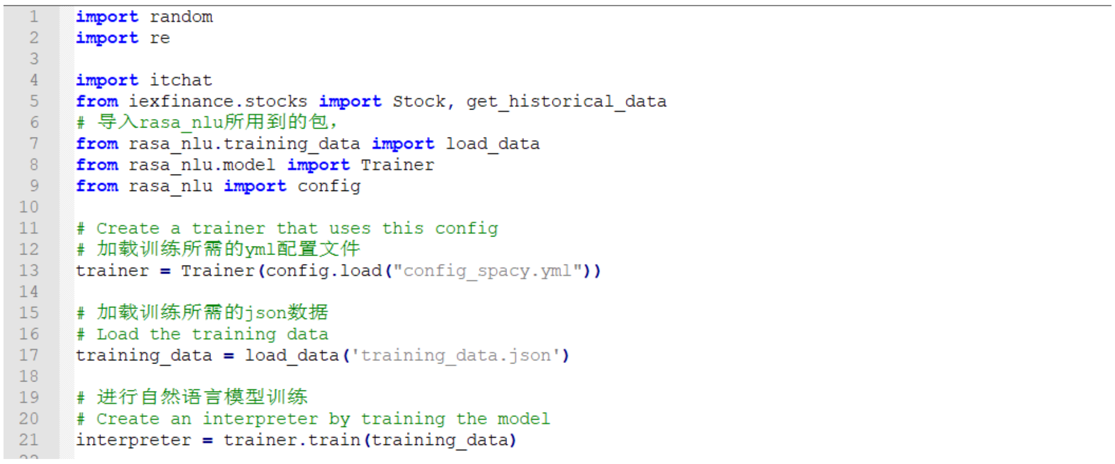
This time I made a stock inquiry robot that can query four aspects of stocks of listed companies: stock price, opening price, closing price and volume. This program can be queried in a single query or multiple rounds.

My programming idea is this: First, through the itchat open the interface between WeChat and the program, to achieve the login state.



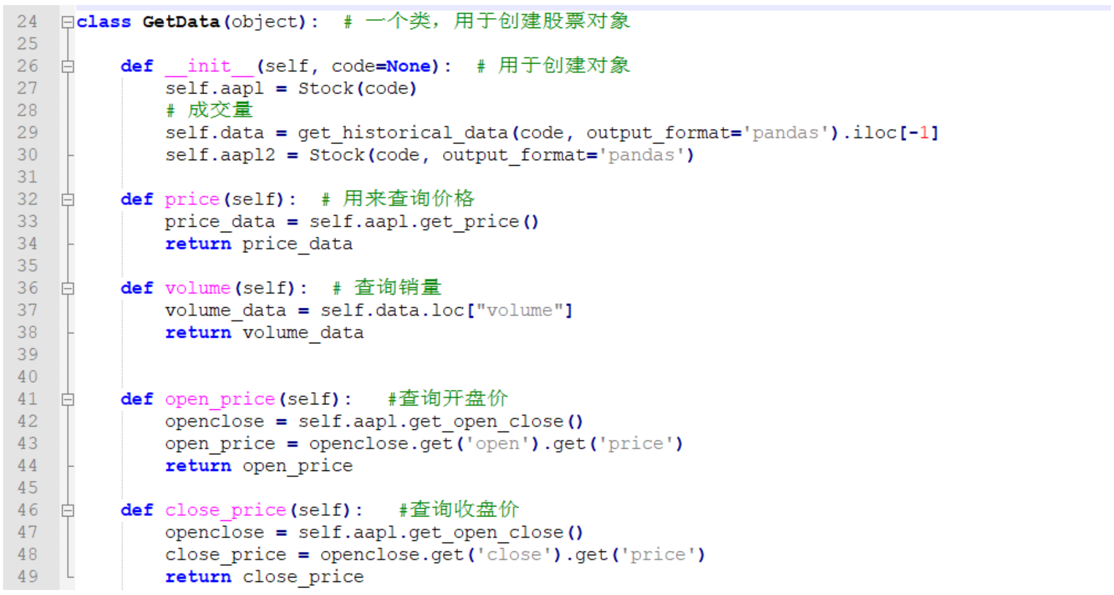
```python

# 执行当前文件
if __name__ == '__main__':
    itchat.auto_login()  # 用于微信登陆、接受信息
    friends = itchat.get_friends(update=True)[0:]
    Name = {}
    Nic = []
    User = []
    for i in range(len(friends)):
        Nic.append(friends[i]["NickName"])
        User.append(friends[i]["UserName"])
    for i in range(len(friends)):
        Name[Nic[i]] = User[i]
    itchat.run()  # 执行之后，弹出二维码

```

Then, open the interface of the iexfinance stock query to realize four types of information such as stock price, volume, opening price and closing price. And the rasa_nlu file is loaded and the natural language model is trained so that the program can recognize the natural language and extract the entity and the intent.



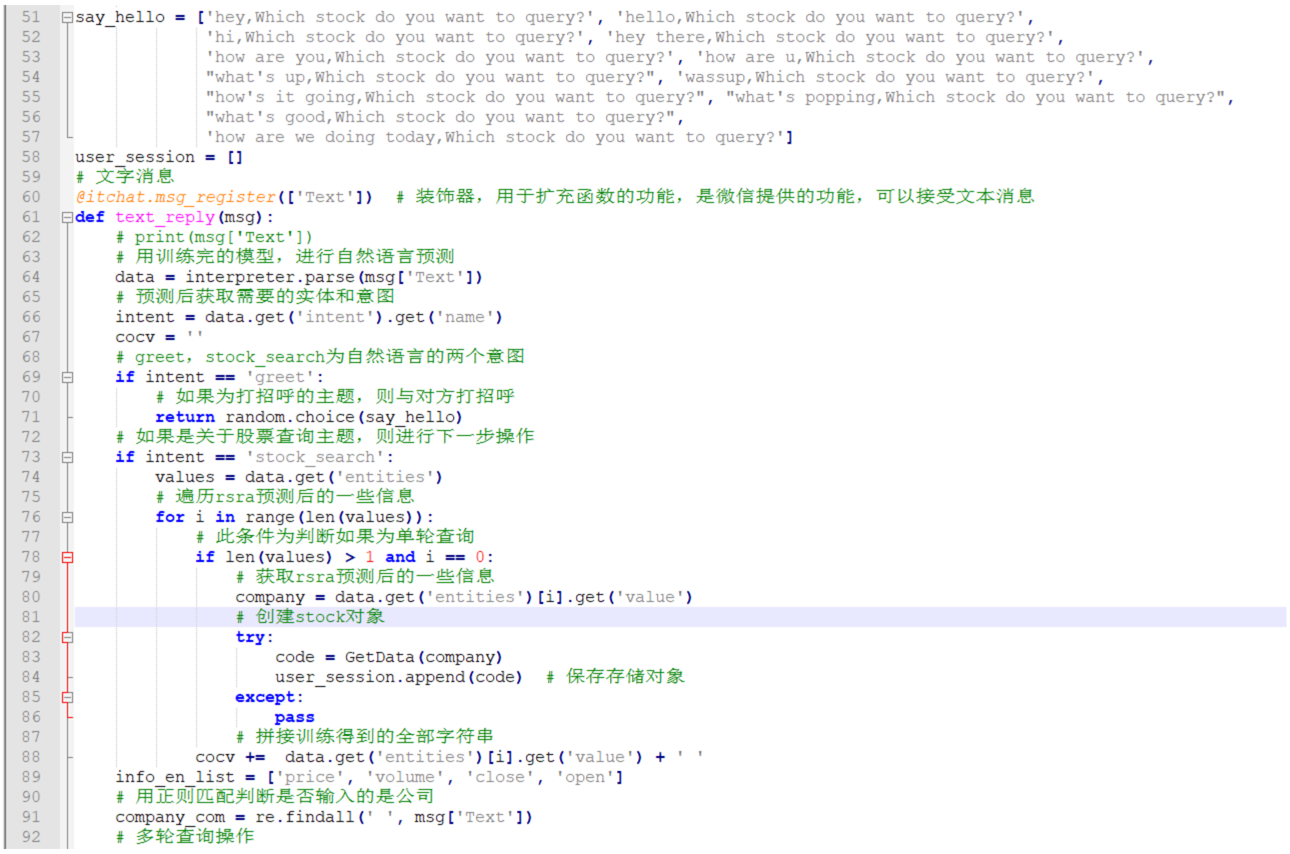
```python
import random
import re

import itchat
from iexfinance.stocks import Stock, get_historical_data
# 导入rasa_nlu所用到的包，
from rasa_nlu.training_data import load_data
from rasa_nlu.model import Trainer
from rasa_nlu import config

# Create a trainer that uses this config
# 加载训练所需的yml配置文件
trainer = Trainer(config.load("config_spacy.yml"))

# 加载训练所需的json数据
# Load the training data
training_data = load_data('training_data.json')

# 进行自然语言模型训练
# Create an interpreter by training the model
interpreter = trainer.train(training_data)
```

After that, we create four objects, which can query the stock price, volume, opening price and closing price.

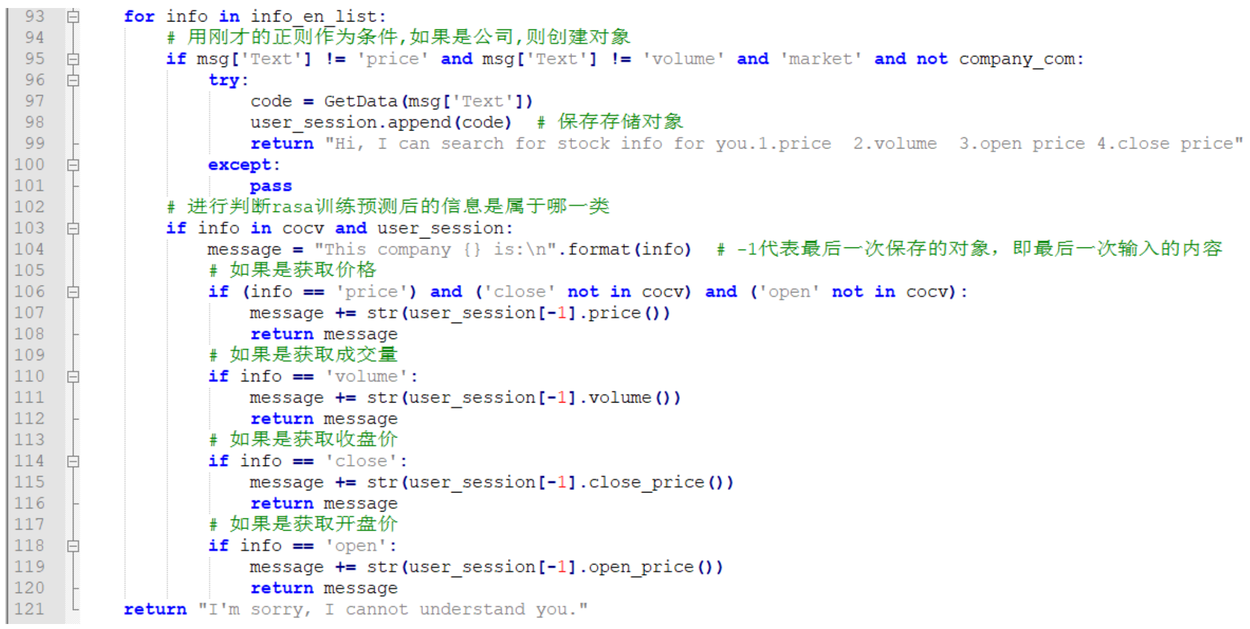


```python
class GetData(object):  # 一个类，用于创建股票对象

    def __init__(self, code=None):  # 用于创建对象
        self.aapl = Stock(code)
        # 成交量
        self.data = get_historical_data(code, output_format='pandas').iloc[-1]
        self.aapl2 = Stock(code, output_format='pandas')

    def price(self):  # 用来查询价格
        price_data = self.aapl.get_price()
        return price_data

    def volume(self):  # 查询销量
        volume_data = self.data.loc["volume"]
        return volume_data


    def open_price(self):   #查询开盘价
        openclose = self.aapl.get_open_close()
        open_price = openclose.get('open').get('price')
        return open_price

    def close_price(self):   #查询收盘价
        openclose = self.aapl.get_open_close()
        close_price = openclose.get('close').get('price')
        return close_price
```

After that, the itchat decorator is used to accept the message sent by the user, and the trained model is used for natural language prediction. Get the required entities and intents after forecasting. If you are greeting the subject, say hello to the other party. If it's about the stock query subject, go ahead. Use rasa_nlu to extract entities and intents, then use regular expressions to determine if the user entered the name of the company. If the information input by the user is the company name, it will ask what content needs to be queried, and then query the information of the stock that the user wants to know according to the message sent by the user. When this program runs to the 76th and 93rd lines, it will judge whether the program is a single round or multiple rounds. If the object is successfully created, the session state will be saved, which will facilitate the subsequent query of other information of the stock. By encapsulating the stock query class, the four types of content such as stock price, volume, opening price and closing price will be realized.



```
51  say_hello = ['hey,Which stock do you want to query?', 'hello,Which stock do you want to query?',
52               'hi,Which stock do you want to query?', 'hey there,Which stock do you want to query?',
53               'how are you,Which stock do you want to query?', 'how are u,Which stock do you want to query?',
54               "what's up,Which stock do you want to query?", 'wassup,Which stock do you want to query?',
55               "how's it going,Which stock do you want to query?", "what's popping,Which stock do you want to query?",
56               "what's good,Which stock do you want to query?",
57               'how are we doing today,Which stock do you want to query?']
58  user_session = []
59  # 文字消息
60  @itchat.msg_register(['Text'])  # 装饰器，用于扩充函数的功能，是微信提供的功能，可以接受文本消息
61  def text_reply(msg):
62      # print(msg['Text'])
63      # 用训练完的模型，进行自然语言预测
64      data = interpreter.parse(msg['Text'])
65      # 预测后获取需要的实体和意图
66      intent = data.get('intent').get('name')
67      cocv = ''
68      # greet，stock_search为自然语言的两个意图
69      if intent == 'greet':
70          # 如果为打招呼的主题，则与对方打招呼
71          return random.choice(say_hello)
72      # 如果是关于股票查询主题，则进行下一步操作
73      if intent == 'stock_search':
74          values = data.get('entities')
75          # 遍历rsra预测后的一些信息
76          for i in range(len(values)):
77              # 此条件为判断如果为单轮查询
78              if len(values) > 1 and i == 0:
79                  # 获取rsra预测后的一些信息
80                  company = data.get('entities')[i].get('value')
81                  # 创建stock对象
82                  try:
83                      code = GetData(company)
84                      user_session.append(code)  # 保存存储对象
85                  except:
86                      pass
87                  # 拼接训练得到的全部字符串
88              cocv +=  data.get('entities')[i].get('value') + ' '
89      info_en_list = ['price', 'volume', 'close', 'open']
90      # 用正则匹配判断是否输入的是公司
91      company_com = re.findall(' ', msg['Text'])
92      # 多轮查询操作
```



```python
    for info in info_en_list:
        # 用刚才的正则作为条件,如果是公司,则创建对象
        if msg['Text'] != 'price' and msg['Text'] != 'volume' and 'market' and not company_com:
            try:
                code = GetData(msg['Text'])
                user_session.append(code)  # 保存存储对象
                return "Hi, I can search for stock info for you.1.price  2.volume  3.open price 4.close price"
            except:
                pass
        # 进行判断rasa训练预测后的信息是属于哪一类
        if info in cocv and user_session:
            message = "This company {} is:\n".format(info)  # -1代表最后一次保存的对象, 即最后一次输入的内容
            # 如果是获取价格
            if (info == 'price') and ('close' not in cocv) and ('open' not in cocv):
                message += str(user_session[-1].price())
                return message
            # 如果是获取成交量
            if info == 'volume':
                message += str(user_session[-1].volume())
                return message
            # 如果是获取收盘价
            if info == 'close':
                message += str(user_session[-1].close_price())
                return message
            # 如果是获取开盘价
            if info == 'open':
                message += str(user_session[-1].open_price())
                return message
    return "I'm sorry, I cannot understand you."
```

When using this program, we need to install the dependency package (equivalent to the program written by someone else, I use it to call, rely on the number in the package to represent the version), then run the written program, after running the program, it will pop up two Dimension, we use WeChat to scan this QR code, which is equivalent to let the current account have the function of automatically querying stocks. After receiving the query information of others, it will automatically query and reply.

At last, I am really grateful to Mr. Fan who gave me the guidance. I learned python in a short time and learned how to write artificial intelligence from him. Thank you very much!

CHUSHU CHEN
2019.5.15