0

远程科研总结报告

王宇桁

2019.8.9

一、 研究背景

1.1 人工智能背景

随着计算机和人工智能技术的迅速发展和普及应用，各行各也都将目光投向了这个新兴的蓬勃发展的方向。然而，任何技术只有将其与我们人类社会的生产生活结合起来，才能最大化这项技术的价值。因此，寻求一种将人工智能和人们的日常生活所结合起来的背景与方向成了越来越多的企业以及高校的研究者们所努力的目标。

那么到底什么是人工智能呢？百度百科的定义是这样的：人工智能是计算机科学的一个分支，它企图了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器，该领域的研究包括机器人、语言识别、图像识别、自然语言处理和专家系统等。人工智能从诞生以来，理论和技术日益成熟，应用领域也不断扩大，可以设想，未来人工智能带来的科技产品，将会是人类智慧的“容器”。人工智能可以对人的意识、思维的信息过程的模拟。人工智能不是人的智能，但能像人那样思考、也可能超过人的智能。

然而正如同其不可估量的发展前景一般，人工智能研究的困难也不可小觑。人工智能是一门极富挑战性的科学，从事这项工作的人必须懂得计算机知识，心理学和哲学。人工智能是包括十分广泛的科学，它由不同的领域组成，如机器学习，计算机视觉等等，总的说来，人工智能研究的一个主要目标是使机器能够胜任一些通常需要人类智能才能完成的复杂工作。但不同的时代、不同的人对这

种“复杂工作”的理解是不同的。[1] 2017 年 12 月，人工智能入选“2017 年度中国媒体十大流行语”。

1.2 项目简介

设计一款以询问股价信息为应用背景的金融聊天机器人。

要求：至少能够实现三种意图的查询（比如问询某几种股票的股价信息、成交量信息、市值等）

1.3 使用工具介绍

为了完成本次项目，我主要采用了如下的几种工具：

①通过 iexfinance 的 API 获取相关的股票信息。

②使用 Rasa NLU，构建训练模型。

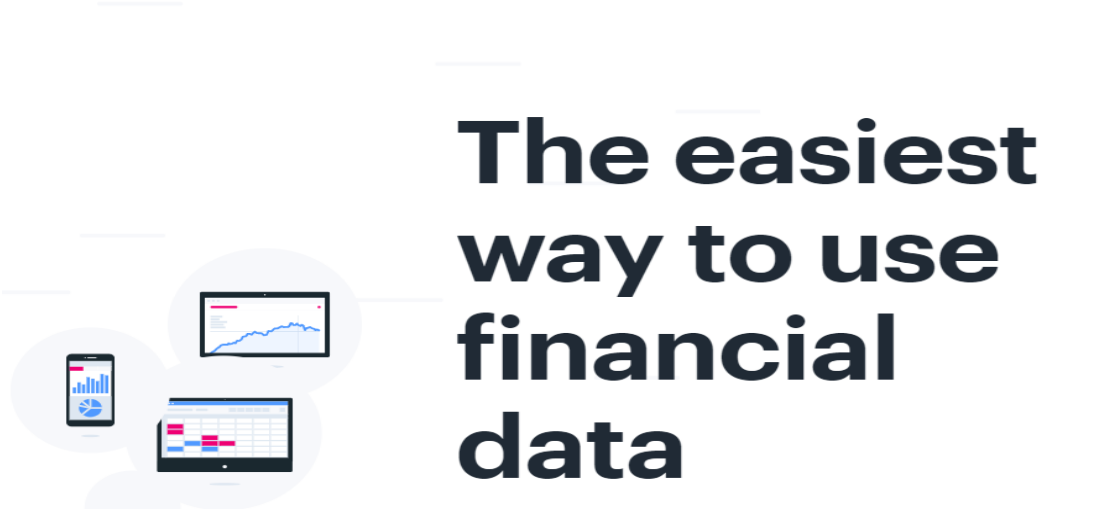
③使用 wxpy 库将完成的 python 代码部署到微信上，最终实现对话机器人。

二、 研究过程及成果

2.1 数据的获取

为了完成项目，首先需要进行的就是相关数据的获取。由于我的本次项目为设计询问股价信息的聊天机器人，因此最为关键的就是获取股票的相关信息。

为此，我选择了 iefinance 的 API 用于获取股票信息。



其中，我采用了 curl 的方式，通过地址调取了股票信息

Curl quote for Apple from the command line

```
curl -k 'https://cloud.iexapis.com/stable/stock/aapl/quote?token=YOUR_TOKEN_HERE'
```

这样就可以发挥其股票的相关信息

```
{'symbol': 'AAPL', 'companyName': 'Apple, Inc.', 'primaryExchange': 'NASDAQ',


'calculationPrice': 'close', 'open': 201.23, 'openTime': 1565357400621,
```

2.2 训练集搭建

为了能够识别说话时人的意图，并且能够提取出信息中“股票的”名字，就需要搭建数量足够大的训练集用于机器人的训练。

训练集中主要包括了非常多机器人可能会遇到的待处理的信息，例如“hi”、“what can you do”、“goodbye”等等。在训练集中，每一个信息包括三个部分：文本信息“text”，包含股票名字的“entities”，以及每句话的意图“intent”，大致结构如图：

```
{
  "text": "i am looking for something about pt",
  "intent": "stock_search",
  "entities": [
      {
      "start": 33,
      "end": 35,
      "value": "pt",
      "entity": "stock"
      }
  ]
},
{
  "text": "search for ab",
  "intent": "stock_search",
  "entities": [
      {
      "start": 11,
      "end": 13,
      "value": "ab",
      "entity": "stock"
      }
  ]
},
{
  "text": "tell me something about ab",
  "intent": "stock_search",
  "entities": [
      {
      "start": 24,
      "end": 26,
      "value": "ab",
      "entity": "stock"
      }
  ]
},
```


其中可见“entities”中包含了股票名字符串的起始与结束位置以及股票的名字和类型，这样就可以准确的提取到其中股票的名字。

而为了能完全符合用户的期望，进行相应的准确回复，非常重要的一点就是能够准确的识别多种意图，并分别进行回复。

为了能够准确识别意图，我将消息的意图分为了

“greet”、“goodbye”、“first_ask”、“stock_search”、“change_ask”、“number”、“buy_stock”、“sell_stock”等八个主要类别。


```
{
  "text": "ok",
  "intent": "affirm",
  "entities": []
},
{
  "text": "great",
  "intent": "affirm",
  "entities": []
},
{
  "text": "hey",
  "intent": "greet",
    "entities": []
},
{
  "text": "howdy",
  "intent": "greet",
    "entities": []
},
{
  "text": "hey there",
  "intent": "greet",
    "entities": []
},
```


2.3 程序设计

在准备好数据以及充分构建好训练集之后，就需要进行代码的实现，其中我的代码主要分为了一下几个模块：

（1） 关键库以及训练数据的调用

程序设计的第一步是声明调用相应的库，以及训练集的调从而使得后续的程序设计能够正常的进行。

```
# 关键库引用
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
import requests
import random

# 引用训练集
trainer = Trainer(config.load("config_spacy.yml"))
training_data = load_data('demo-rasa-noents.json')
interpreter = trainer.train(training_data)
```

完成之后，之后就能够成功识别语句的意图，以“hello”为例，其效果如图所示：

```
{'intent': {'name': 'greet', 'confidence': 0.7450154861591415}, 'entities': [],
```

(2) 股票名称提取与股票信息获取

接下来需要设计函数用于提取语句中可能会包含的股票的名字，并且设计函数用于利用 api 对相关股票信息的获取。

```python
# 股票名称获取
def get_stock(message):
  entities = interpreter.parse(message)["entities"] #实体获取
  params = {}
  if len(entities)!=0:
     for ent in entities:
       params[ent["entity"]] = str(ent["value"])
     return params['stock']
  else:
      return None

# 股票信息获取
def stock_info(stock, type):
    start = 'https://cloud.iexapis.com/stable/stock/'
    end = '/quote?token=pk_5b4b75ef0e6c4d6c9bbf32925604b841'
    Stock = stock
    r = requests.get(start + Stock + end)#api获取数据
    dic = r.json()
    return dic[type]
```

（3） 状态转换字典

为了能够根据不同的语句意图进行不同的回答，并且能够根据不同的语句进行不同的回答，并且回答也需要符合一定的语序，使用状态转换的方式是能够实现的一种方法。其中，最重要的一点就是设计状态转换的字典，使得状态能够成转换。

```
INIT=0   # 状态定义
AUTHED=1
CHOOSE_STOCK=2
ASKING=3
AUTHED2=4
BUYING=5
SELLING=6
ORDERED=7
TRADE=8

policy_rules = {
    (INIT, "greet"):(INIT, random.choice(keywords['greet']), None),
    (INIT, "goodbye"): (INIT,random.choice(keywords['goodbye']), None),
    (INIT, "first_ask"): (INIT, "Hi, i'm a chatbot to help you get information about stock market",None),
    (INIT, "stock_search"): (INIT, "Sorry :(, you have to log in first, may i have your phone number please?", AUTHED),
    (INIT, "number"): (AUTHED, "Welcome back!",None),
    (AUTHED, "stock_search"): (ASKING, "Well, which stock are you interested in?", None),
    (ASKING, "stock_search"): (ASKING, "What information do you want to know?", None),
    (ASKING, "latestprice_search"): (ASKING, "Got it! Its latest price is", None),
    (ASKING, "latestvolume_search"): (ASKING, "Well, its latest price is", None),
    (ASKING, "week52high_search"): (ASKING, "OK, its highest price in 52 weeks is", None),
    (ASKING, "change_ask"): (ASKING, "Sure, what else do you want to know", None),
    (AUTHED, "goodbye"): (INIT,random.choice(keywords['goodbye']), None),
    (ASKING, "goodbye"): (INIT,random.choice(keywords['goodbye']), None),
    (ASKING, "buy_stock"): (ASKING,"No problem, may i have your account please?", AUTHED2),
    (ASKING, "sell_stock"): (ASKING,"No problem, may i have your account please?", AUTHED2),
    (ASKING, "number"): (AUTHED2, "Welcome back!", None),
    (AUTHED2, "buy_stock"): (BUYING, "So how many do you want to buy?", None),
    (AUTHED2, "sell_stock"): (SELLING, "So how many do you want to sell?",None),
    (BUYING, "number"): (ORDERED, "Alright, the trade is finished, remember to check your account :)", None),
    (SELLING, "number"):(ORDERED, "Alright, the trade is finished, remember to check your account :)", None),
    (ORDERED, "goodbye"): (INIT,random.choice(keywords['goodbye']), None),
    (ORDERED, "stock_search"):(ASKING, "What information do you want to know?", None),
    (ORDERED, "buy_stock"): (BUYING, "So how many do you want to buy?",None),
    (ORDERED, "sell_stock"):(SELLING, "So how many do you want to sell?", None)

}
```

（4） 回复函数

紧接着需要设计回复函数。回复函数的主要目的是利用传入的状态等参数，根据状态转换字典的规定对状态等参数进行刷新，并且将回复的内容存储在 list 当中，最后返回刷新后的状态等参数，用于之后的信息回复。

```
# 回复函数建立
stock_all=[]


def send_message(state, pending, message, answer):

    if get_stock(message)!=None:            #股票名存储
        stock_all.append(get_stock(message))
    if len(stock_all) == 0:
        stock = None
    else: stock =stock_all[-1]
    data = interpret(message)
    print("USER:{}".format(message))
    new_state, response, pending_state = policy_rules[(state, interpret(message)["intent"]["name"])]
    print("BOT:{}".format(response))
    answer.append(response)   #回复消息存储


    if pending is not None:
        new_state, response, pending_state = policy_rules[pending]
        print("BOT:{}".format(response))
        answer.append(response)
        pending = None
    if pending_state is not None:
        pending = (pending_state, interpret(message)["intent"]["name"])
    if (new_state == ASKING) & (interpret(message)["intent"]["name"]=="latestprice_search"):
        print("BOT:{}".format(stock_info(stock, 'latestPrice')))
        answer.append(stock_info(stock, 'latestPrice'))
    if (new_state == ASKING) & (interpret(message)["intent"]["name"]=="latestvolume_search"):
        print("BOT:{}".format(stock_info(stock, 'latestVolume')))
        answer.append(stock_info(stock, 'latestVolume'))
    if (new_state == ASKING) & (interpret(message)["intent"]["name"]=="week52high_search"):
        print("BOT:{}".format(stock_info(stock, 'week52High')))
        answer.append(stock_info(stock, 'week52High'))
    return new_state, pending
```

（5） 总体回复函数

总体的回复函数则是基于已完成的回复函数，根据用户输入的语句，进行状态的更新，从而使得回复函数能够顺利地对不同的语句进行回复。

```
# 总体回复函数建立
state = INIT  #全局变量建立
pending = None
answer = []

def send_messages(messages):
    global state #通过函数改变全局变量
    global pending
    global answer
    answer = []
    state, pending = send_message(state, pending, messages, answer)
```

2.4 微信集成

为了能够使得对话机器人更加贴近生活，我选择使用 wxpy 库将

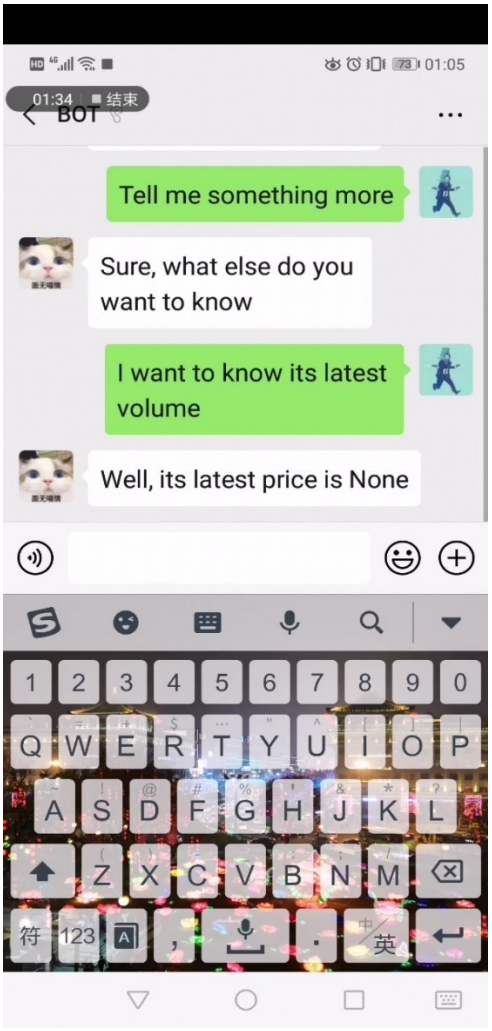
已经设计好的函数集成到微信上，其函数大致如下：

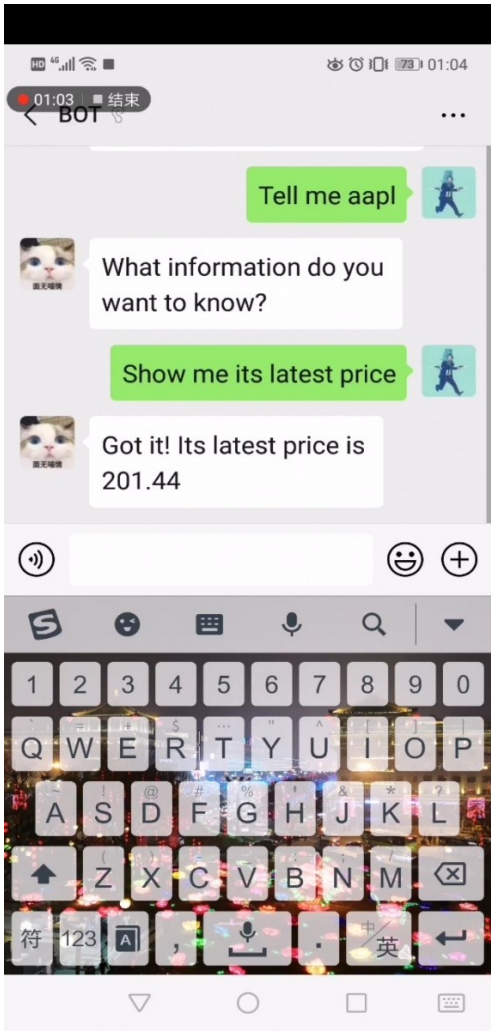
```
#微信集成代码
from wxpy import *

bot = Bot(cache_path=True) #定义机器人


sender = bot.search('王宇桁')[0] #特定对象获取


@bot.register()  # 用于注册消息配置
def recv_send_msg(recv_msg):
    print('收到的消息：', recv_msg.text)  # recv_msg.text取得文本
    # recv_msg.sender 就是谁给我发的消息这个人
    if recv_msg.sender == sender:
        print(stock_info("aapl", 'latestPrice'))
        send_messages(recv_msg.text)
        print(len(answer))
        if len(answer) == 1: #回复只有一句则直接回复，两句以上则合并后回复
          ms = answer[-1]
          print(ms)
        if len(answer) >= 2:
          ms = answer[-2] + ' ' +str(answer[-1])
    return ms  # 回复


embed() # 进入交互式的 Python 命令行界面，并堵塞当前线程支持使用 ipython, bpython 以及原生 python
```

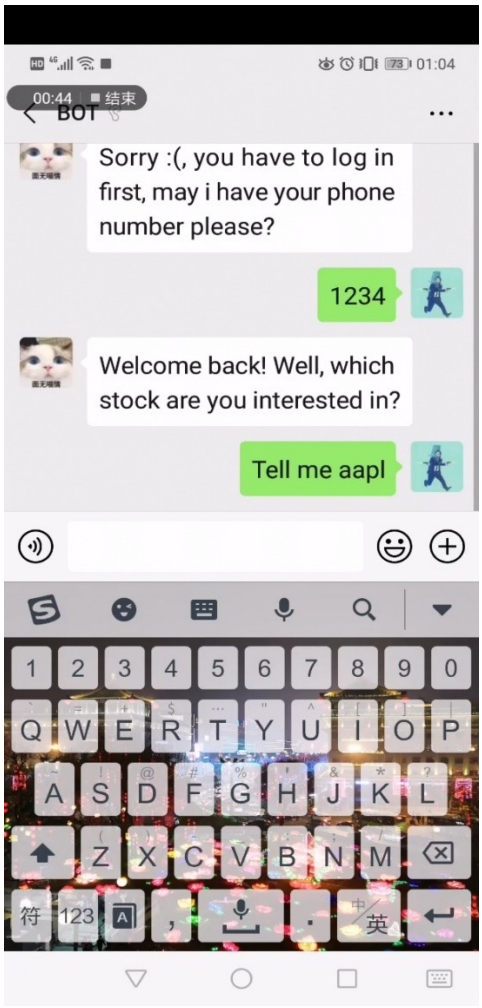
其中包括了以下几个步骤：获取接收到的消息，获取消息的来源，对获取的消息进行回复，将回复内容重新发给信息的来源，最终成功将其与微信绑定在一起。

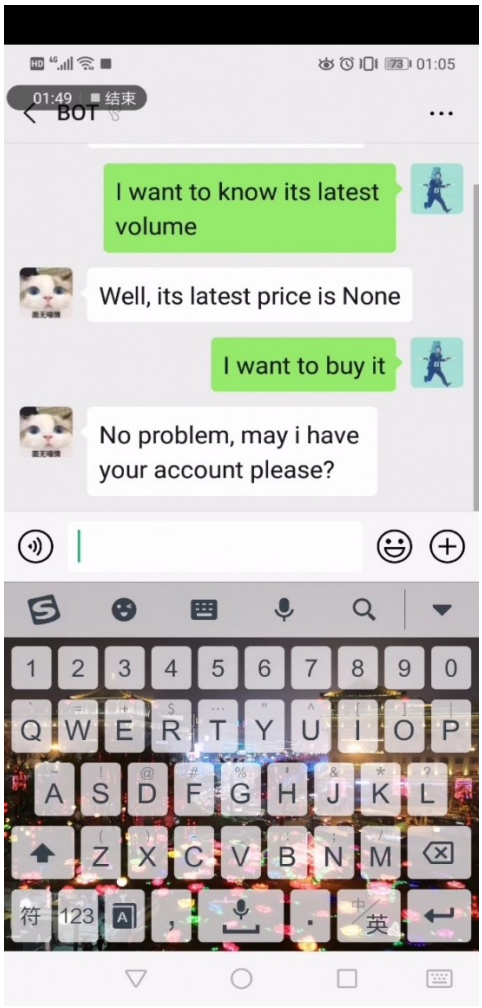
2.5 实验结果

以上步骤完成之后，可以通过微信与机器人进行对话，其结果如下：


01:05
01:34 结束
BOT
Tell me something more
Sure, what else do you want to know
I want to know its latest volume
Well, its latest price is None


01:04
01:03 结束
BOT
Tell me aapl
What information do you want to know?
Show me its latest price
Got it! Its latest price is 201.44





三、 学习感受与收获

在报名参加本次远程科研项目之前，“人工智能”一词对我来说只是一个模糊的概念。我对此唯一了解的就是它是一个飞速发展中的学科，越来越多的人将研究的重心向这门学科靠拢，但是对于它实际在我们的日常生活中能起到什么样的作用却是是知之甚少。

通过这次的科研项目，我亲自利用上课中老师所介绍的方法设计了设计了一款能够识别用户意图的股票信息查询的对话机器人，我深切体会到人工智能将来在我们日常生活中所扮演的角色有多么的重要。

在四周的学习过程中，老师给我们讲解了许多可以识别人类自然语言意图，并对于其进行相应的回答的方法。比如通过正则表达式、模式匹配、关键词提取、句法转换等来回答问题；过正则表达式、最近邻分类法或者支持向量机之一或多种方案来提取用户意图。同时老师也教会了我们使用 Rasa NLU 这个可以用于自然语言识别的工具，我学会了如何构建训练集来对机器人“训练”，使得其能够像正常人类一样识别语言中意图。虽然只有四节课的内容，但是每节课上老师都准备了充分的代码练习，使我们能够边学边练。虽然每节课上的内容第一次看都感觉问题不大，但是在自己独自课后复习上课内容时总会遇到许许多多的问题，因此需要不断地复习和巩固，并且积极地独自思考。是在遇到无法解决的问题时，我会向老师求助，老师能够耐心地解决我的很多问题。在与老师的交流中，我总会发现一些代码中漏洞，最后成功的弥补上。最后我完成对话机器人的时候，感觉到十分的激动。

这次的科研项目不仅让我获得了平时在课堂上无法掌握的知识，更激发了我对于人工智能这个方向的兴趣。在参加项目之前，我从未亲自参与一个和人工智能有关的项目。通过这次的项目，我愈加了解到人工智能这门学科的魅力，我了解到人工智能不仅能够为我们带来乐趣，同样能够大幅度简化我们日常生活中的一些比较繁琐的工作。我认为作为一名计算机相关专业的学生，我将来应当更加重视这方面的学习，只有这样才能不断地丰富自己，以满足社会未来发展所带来的不断增长的人才需求。