MIT 金融聊天机器人项目报告

报告人：李新璋

学校：Virginia Tech

日期：12/15/2018

目 录

1. 背景介绍

随着人工智能的时代到来，聊天机器人越来越多的运用在各行各业，有的甚至连淘宝的客服都会有聊天机器人的参与。聊天机器人并不只是用来和人聊天，更多时候是用于解决特定的问题而生的。比如说像我刚刚提到的淘宝客服机器人，会去对客户发来的对话做一个简单的理解，查看多方想要干什么，比如 查看快递单号，追踪快递，或者再有就是人工智能解决不了的问题，就将其转入人工服务。对于聊天机器人的定义，维基百科是这样说的： 聊天机器人是经由对话或文字进行交谈的计算机程序。能够模拟人类对话。聊天机器人可用于实用的目的，如客户服务或资讯获取。有些聊天机器人会搭载自然语言处理系统，但大多简单的系统只会撷取输入的关键字，再从数据库中找寻最合适的应答句。目前，聊天机器人是虚拟助理（如 Google 智能助理）的一部分，可以与许多组织的应用程序，网站以及即时消息平台（Facebook Messenger）连接。非助理应用程序包括娱乐目的的聊天室，研究和特定产品促销，社交机器人。

2. 项目简介

这次项目是在接受 AI 远程课程授课 4 次授课之后完成的项目。具体项目为采用问询股票信息为应用背景的金融聊天机器人。使用 Rasa NLU 进行项目的数据训练，然后判断意图。使用 IEXFinance API 进行比如股票价格，开盘价格等数据的请求并处理返回 JSON 值。最后用过 WXPY，把聊天机器人部署到微信上面。

3. 项目分析

当得到任务之后我将其分成了四个部分：数据训练，编写多轮多次查询，调取 IEXFinance API 获得金融数据，使用 WXPY 进行微信整合。

首先我来回顾一下课程里面学了什么：

1. 同一个问题多种选择性的回答，并提供缺省回答的方案；
2. 能通过正则表达式、模式匹配、关键词提取、句法转换等来回答问题；
3. 能通过正则表达式、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；
4. 通过预建的命名实体类型、角色关系、依赖分析等来进行命名实体识别；
5. 基于 Rasa NLU 的本地基础聊天机器人系统的构建；
6. 数据库查询并使用自然语言探索数据库内容（提取参数、创建查询、响应）
7. 基于增量过滤器的单轮多次增量查询技术以及甄别否定实体技术
8. 实现状态机的多轮多次查询技术，并能基于语境问题提供解释和回答
9. 处理拒绝、等待状态转换和待定行动的多轮多次查询技术

首先第一步就是使用学习的第五项，基于 Rasa NLU 进行本地基础聊天机器人系统构建。我根据课程里面提供的例子进行了改写完成了这个部分。然后就是编写多轮多次了查询了。首先就是设置一个大概的

policy 和基本的状态机状态如下图代码截图所示

```
def policyrule():
    global stock_re
    policy = {
        (INIT, "none"): (INIT, "I'm sorry - I'm not sure how to help you"),
        (INIT, "greet"): (PENDING, "Hello. I'm a stock inquiry bot. You can ask questions about the stock of a specific company"),
        (INIT, "thankyou"): (THANK, "You are very welcome"),
        (INIT, "stock_search"): (CHOOSED, stock_re),

        (PENDING, "none"): (PENDING, "I'm sorry, which stock do you wanna look at?"),
        (PENDING, "greet"): (PENDING, "I'm sorry, which stock do you wanna look at?"),
        (PENDING, "thankyou"): (PENDING, "I'm sorry, which stock do you wanna look at?"),
        (PENDING, "stock_search"): (CHOOSED, stock_re),

        (CHOOSED, "none"): (CHOOSED, "I'm sorry, which stock do you wanna look at?"),
        (CHOOSED, "greet"): (CHOOSED, "I'm sorry, which stock do you wanna look at?"),
        (CHOOSED, "thankyou"): (THANK, "You are very welcome"),
        (CHOOSED, "stock_search"): (CHOOSED, stock_re),

        (THANK, "none"): (INIT, "I'm sorry - I'm not sure how to help you"),
        (THANK, "greet"): (PENDING, "Hello. I'm a stock inquiry bot. You can ask questions about the stock of a specific company"),
        (THANK, "thankyou"): (THANK, "You are very welcome"),
        (THANK, "stock_search"): (CHOOSED, stock_re),
    }
    return policy
```

```
# Define the states
INIT = 0
PENDING = 1
CHOOSED = 2
THANK = 3
```

在定义完基础 policy 之后就是编写回答和调取 IEXFinance API，代码如下图截图所示

```
def stock_return(message):
    result = interpreter.parse(message)

    global stock_re
    global setting

    if 'open' in result.get('entities')[0].get('value'):
        company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        open_price = Stock(company).get_open()
        stock_re = "The open price is {}".format(open_price)

    if 'close' in result.get('entities')[0].get('value'):
        company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        close_price = Stock(company).get_price()
        stock_re = "The close price is {}".format(close_price)

    if 'volume' in result.get('entities')[0].get('value'):
        company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        volume = Stock(company).get_volume()
        stock_re = "The volume is {}".format(volume)
```

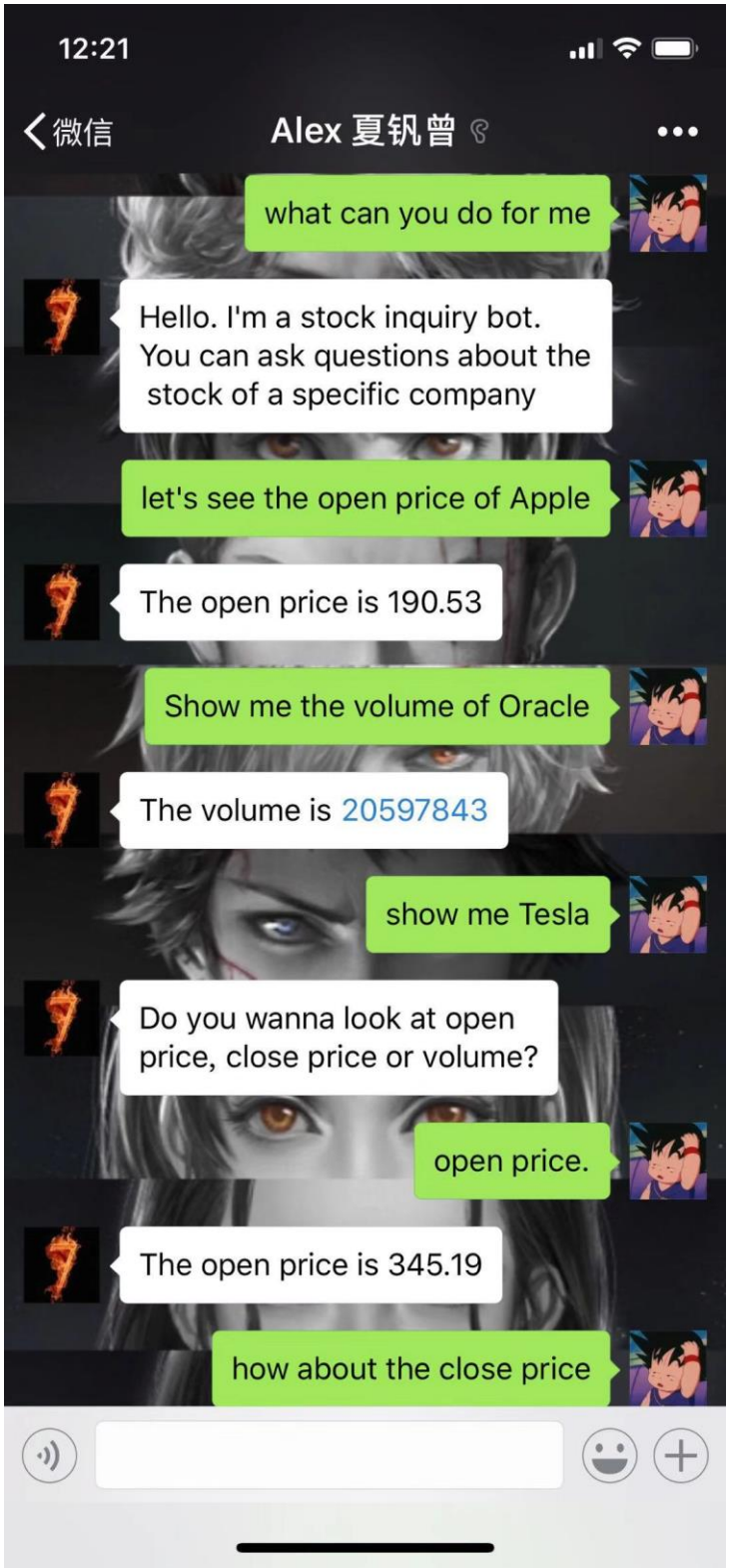
```
def stock_pending(message):
    result = interpreter.parse(message)

    global stock_re
    global remember
    global flag
    global setting

    company = ''

    if 'open' in result.get('entities')[0].get('value'):
        if remember != '':
            company = remember
        else:
            company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        open_price = Stock(company).get_open()
        stock_re = "The open price is {}".format(open_price)

    if 'close' in result.get('entities')[0].get('value'):
        if remember != '':
            company = remember
        else:
            company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        close_price = Stock(company).get_price()
        stock_re = "The close price is {}".format(close_price)

    if 'volume' in result.get('entities')[0].get('value'):
        if remember != '':
            company = remember
        else:
            company = result.get('entities')[1].get('value')
        for i in setting:
            if company in i.get('name'):
                company = i.get('symbol')
        volume = Stock(company).get_volume()
        stock_re = "The volume is {}".format(volume)

    if result.get('entities')[0].get('entity') == 'company':
        stock_re = "Do you wanna look at open price, close price or volume?"
        remember = result.get('entities')[0].get('value')
        for i in setting:
            if remember in i.get('name'):
                remember = i.get('symbol')

    flag = True
```

最后就是使用 WXPY 进行微信的整合

```
from wxpy import *

bot = Bot(cache_path=True)

my_friend = bot.friends().search('Shao')[0]

@bot.register(my_friend)
def my_friednd_message(msg):
    global state
    global response
    message = str(msg.text)
    state = send_message(policyrule(), state, message)
    return response
```

4. 总结成果

最后的聊天机器人成品效果图如下



在这次聊天机器人课程的学习中遇到了很多困难，同时也学到了很多

有意思的知识，让我对基本人工智能 AI 有了一个大致的了解，还想通过 TensorFlow 进行对现有机器人进行优化。总之就是真的学到了很多，让我对人工智能产生了兴趣。