智能聊天机器人课程总结

姓名:许瑞琪

时间：2018.8.7-2018.8.28

目录

一、 问题研究背景及现状

聊天机器人（chatbot）是一个用来模拟人类对话或聊天的程序。聊天机器人产生的原因是，研发者把自己感兴趣的回答放到数据库中，当一个问题被抛给聊天机器人时，它通过算法，从数据库中找到最贴切的答案，回复给它的聊伴。

目前市面上主要的智能聊天机器人可以分为如下两类：目标驱动型聊天机器人和无目标驱动型聊天机器人。目标驱动机器人是指机器人的服务目标或服务对象是明确的，是可以提供特殊服务的问答系统，处理特定领域的问题，即定领域的聊天机器人，比如客服机器人，订票机器人等。无目标驱动机器人是指机器人的服务对象和聊天范围不明确，可以处理的问题多种多样，解决问题时需要依赖各种信息和本体，即开放领域的聊天机器人，比如娱乐聊天机器人等。

智能聊天机器人实际上是为了应对信息爆炸的今天存在的信息过载问题。具体来说，其来源是因为人们对于简单的搜索引擎仅仅返回一个网页集合的不满，而通常用户更想获得的体验是在向智能对话系统用自然语言提出一个问题之后，且智能对话系统也能够自然又通顺地回答问题，且回答内容与问题紧凑相关又答案精准。为使用者们节约了更多的时间，无需逐个浏览和仔细阅读搜索引擎返回的每个链接网址中的信息，再剔除冗余信息后才能得到期望的答案。

有关于对话机器人的研究可以被追溯到 20 世纪 50 年代，当 Alan M. Turing 提出了“机器可以思考吗？”的图灵测试问题来衡量人工智能发展的程度，该领域接下来就变成了人工智能领域中一个十分有趣又具有挑战性的研究问题。

随着各种互联网公司的蓬勃发展以及各类移动终端和应用小软件的爆炸式普及，如 Twitter 和 Facebook 等，很多大互联网公司都在投入重金完成此领域技术的研究并陆续推出此类应用产品，比如苹果 Siri，微软 Cortana，脸书 Messenger，谷歌 Assistant，IBM Waston 等，这些产品都让用户们在移动终端更加方便地获得需要的信息和服务，从而获得更好的用户体验。因而，人们发现智能机器人可以应用的领域十分广泛，它可以被应用到很多人机交互的领域中，比如技术问答系统，洽谈协商，电子商务，家教辅导，娱乐闲聊等。

不得不说，由于人们对于智能聊天机器人不断增长的渴望和需求，人工智能在自然语言处理领域的应用变成了不论国内国外都非常热门的一个研究方向。在信息技术飞速发展，以及移动终端逐渐普及的今天，研究聊天机器人相关的技术，对于促进人工智能以及人机交互方式的发展有着十分重大的意义。

二、 问题研究过程

1. 自然语言理解难点

自人工智能在 1956 年达特茅斯会议上首次提出，让机器完成更多的智力工作成为科学家努力的方向。其中一个重要的目标就是希望机器能够与人类进行更加自然高效的交流，希望机器读懂人类深奥的语言，同时以一种我们习惯的方式进行交互，而解决这个问题的关键技术就是自然语言处理。

NLP 是计算机以一种聪明而有用的方式分析，理解和从人类语言中获取意义的一种方式。通过利用 NLP，开发者可以组织和构建知识来执行词性标注、句法分析、命名实体识别、自然语言理解、关系提取、语音识别等。目前 NLP 的方法是基于深度学习，这是一种 AI，它检查和使用数据中的模式来改善程序的理解。深度学习模型需要大量的标记数据来训练和识别相关的相关性，汇集这种大数据集是当前 NLP 的主要障碍之一。

此外目前自然语言处理领域的难点还有以下几点：

① 单词的边界界定

在口语中，词与词之间通常是连贯的，而界定字词边界通常使用的办法是取用能让给定的上下文最为通顺且在文法上无误的一种最佳组合。在书写上，汉语也没有词与词之间的边界。

② 词义的消歧

许多字词不单只有一个意思，因而我们必须选出使句意最为通顺的解释。

③ 句法的模糊性

自然语言的文法通常是模棱两可的，针对一个句子通常可能会剖析（Parse）出多棵剖析树（Parse Tree），而我们必须要仰赖语义及前后文的信息才能在其中选择一棵最为适合的剖析树。

④ 有瑕疵的或不规范的输入

例如语音处理时遇到地方口音，或者在文本的处理中处理拼写，语法或者光学字符识别（OCR）的错误。

⑤ 语言行为与计划

句子常常并不只是字面上的意思；例如，“你能把盐递过来吗”，一个好的回答应当是动手把盐递过去；在大多数上下文环境中，“能”将是糟糕的回答，虽说回答“不”或者“太远了我拿不到”也是可以接受的。再者，如果一门课程去年没开设，对于提问“这门课程去年有多少学生没通过？”回答“去年没开这门课”要比回答“没人没通过”好。

因此，本次科研营，我们同老师一起了解自然语言处理技术，改善其中存在的问题，使其更好地服务于人类。

2. 研究工具

2.1 spacy

spacy 是一个 Python 自然语言处理工具包，诞生于 2014 年年中，号称“Industrial-Strength Natural Language Processing in Python”，是具有工业级强度的 Python NLP 工具包。

其功能主要有：

① 词性标注(Part-of-speech tagging)

② 依赖分析(Dependency parsing)
③ 命名实体识别(Named entities recognition)
④ 培训和更新(Training and updating)
⑤ 使用正则表达式(Using regular expressions)
⑥ 单词向量与语义相似度(Word Vectors and Semantic Similarity)
⑦ 模型训练(Training Spacy's Statistical Models)

2.2 Rasa_nlu

Rasa NLU 主要功能是用户问句的意图识别和实体抽取。

2.3 Iexfinance

围绕 Investors Exchange（IEX） Developer API 的 Python 包装器 。
一个易于使用的界面，以获得：

- 实时报价
- 历史数据
- 基本面上，
- 行动（股息，分割），行业表现
- 交易分析
- IEX 市场数据和统计

3. 知识点回顾

3.1 意图&实体

意图，对于领域数据的操作，表示用户想要完成的任务，一般以动宾短语来命名；比如金融股票领域中，有“查开盘价”、“查最高价”等意图；

命名实体识别指识别文本中具有特定意义的实体，如人名、机构名、地名等专有名词和有意义的时间等，是信息检索、问答系统等技术的基础任务。如在“I want the price of APPLE”中，命名实体有：“APPLE——organization”。

举个例子，当输入“Tell me the high price of Microsoft”，NLU 模块就可以识别出用户的意图是“get_high”，而关键实体是“Microsoft”。有了意图和关键实体，就方便了后面对话管理模块进行后端数据库的查询或是有缺失信息而来继续多轮对话补全其它缺失的实体槽。

3.2 正则表达式

正则表达式是一种定义了搜索模式的特征序列，主要是用于字符串的模式匹配，或是字符的匹配。

Python 通过 re 模块提供对正则表达式的支持。

使用 re 的一般步骤是

- 1. 将正则表达式的字符串形式编译为 Pattern 实例
- 2. 使用 Pattern 实例处理文本并获得匹配结果（一个 Match 实例）
- 3. 使用 Match 实例获得信息，进行其他的操作。

3.3 意图识别

意图识别（Intent）：在句子级别进行分类，明确意图。

3.4 实体抽取

实体识别（Entity）：在词级别找出用户问题中的关键实体，进行实体槽填充（Slot Filling）。

命名实体识别（NER）：命名实体是命名实体识别的研究主体，一般包括 3 大类（实体类、时间类和数字类）和 7 小类（人名、地名、机构名、时间、日期、货币和百分比）命名实体。

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

```
Apple 0 5 ORG
U.K. 27 31 GPE
$1 billion 44 54 MONEY
```

3.5 词向量与相似度

spaCy 能够比较两个对象，并预测它们的相似程度。预测相似性对于构建推荐系统或标记重复项非常有用。每个 Doc，Span 并 Token 附带一个.similarity() 方法，可以与另一个对象进行比较，并确定相似性。

```
import spacy

nlp = spacy.load('en_core_web_md')  # make sure to use larger model!
tokens = nlp(u'dog cat banana')

for token1 in tokens:
    for token2 in tokens:
        print(token1.text, token2.text, token1.similarity(token2))
```

```
dog dog 1.0
dog cat 0.80168545
dog banana 0.24327643
cat dog 0.80168545
cat cat 1.0
cat banana 0.28154364
banana dog 0.24327643
banana cat 0.28154364
banana banana 1.0
```

3.6 SQLite

不像常见的客户端/服务器结构数据库管理系统，SQLite 引擎不是一个应用程序与之通信的独立进程。SQLite 库链接到程序中，并成为它的一个组成部分。这个库也可被动态链接。应用程序经由编程语言内的直接 API 调用来使用 SQlite 的功能，这在减少数据库访问延迟上有积极作用，因为在一个单一进程中的函数调用比跨进程通信更有效率。SQLite 将整个数据库，包括定义、表、索引以及数据本身，作为一个单独的、可跨平台使用的文件存储在主机中。它采用了在写入数据时将整个数据库文件加锁的简单设计。尽管写操作只能串行进行，但 SQLite 的读操作可以多任务同时进行。

In [3]:
```python
# Import sqlite3
import sqlite3

# Open connection to DB
conn = sqlite3.connect('hotels.db')

# Create a cursor
c = conn.cursor()

# Define area and price
location, price = "south", "hi"
t = (location, price)

# Execute the query
c.execute('SELECT * FROM hotels WHERE location=? AND price=?', t)

# Print the results
print(c.fetchall())
```

```
[('Grand Hotel', 'hi', 'south', 5)]
```

3.7 多轮查询

```
def respond(message, params):
```
#更新参数

3.8 甄别否定实体

```
for end in ends:
    chunks.append(phrase[start:end])
    start = end
result = {}
# Iterate over the chunks and look for entities
for chunk in chunks:
    for ent in ents:
        if ent in chunk:
            # If the entity is preceeded by a negati
            if "not" in chunk or "n't" in chunk:
                result[ent] = False
            else:
                result[ent] = True
return result
```

#查找 not& n't

#查找否定实体进行 chunk 切割

3.9 实现状态机

关于状态机的一个极度确切的描述是它是一个有向图形，由一组节点和一组相应的转移函数组成。状态机通过响应一系列事件而“运行”。每个事件都在属于“当前” 节点的转移函数的控制范围内，其中函数的范围是节点的一个子集。函数返回“下一个”（也许是同一个）节点。这些节点中至少有一个必须是终态。当到达终态， 状态机停止。

状态机可归纳为 4 个要素，即现态、条件、动作、次态。这样的归纳，主要是出于对状态机的内在因果关系的考虑。“现态”和“条件”是因，“动作”和“次态”是果。详解如下：

①现态：是指当前所处的状态。

②条件：又称为“事件”，当一个条件被满足，将会触发一个动作，或者执行一次状态的迁移。

③动作：条件满足后执行的动作。动作执行完毕后，可以迁移到新的状态，也可以仍旧保持原状态。动作不是必需的，当条件满足后，也可以不执行任何动作，直接迁移到新状态。

④次态：条件满足后要迁往的新状态。“次态”是相对于“现态”而言的，“次态”一旦被激活，就转变成新的“现态”了。

```
# Define the INIT state
INIT=0

# Define the CHOOSE_COFFEE state
CHOOSE_COFFEE=1

# Define the ORDERED state
ORDERED=2

# Define the policy rules
policy = {
    (INIT, "order"): (CHOOSE_COFFEE, "ok, Colombian or Kenyan?"),
    (INIT, "none"): (INIT, "I'm sorry - I'm not sure how to help you"),
    (CHOOSE_COFFEE, "specify_coffee"): (ORDERED, "perfect, the beans are on their way!"),
    (CHOOSE_COFFEE, "none"): (CHOOSE_COFFEE, "I'm sorry - would you like Colombian or Kenyan?"),
}

# Create the list of messages
messages = [
    "I'd like to become a professional dancer",
    "well then I'd like to order some coffee",
    "my favourite animal is a zebra",
    "kenyan"
]

# Call send_message() for each message
state = INIT
for message in messages:
    state = send_message(policy, state, message)
```

```
USER : I'd like to become a professional dancer
BOT : I'm sorry - I'm not sure how to help you
USER : well then I'd like to order some coffee
BOT : ok, Colombian or Kenyan?
USER : my favourite animal is a zebra
BOT : I'm sorry - would you like Colombian or Kenyan?
USER : kenyan
BOT : perfect, the beans are on their way!
```

4 构建过程

4.1 processing pipeline

在 Rasa NLU 中，传入消息由一系列组件处理。这些组件在所谓的 processing pipeline 中一个接一个地执行。存在用于实体提取，用于意图分类，预处理等的组件。

spacy_sklearn pipeline 使用来自 GloVe 或 fastText 的预先训练的单词向量。spacy_sklearn pipeline 的优势在于，如果有一个训练样例：“我想买苹果”，并且要求 Rasa 预测“获得梨子”的意图,模型已经知道“苹果”和“梨”这两个词“非常相似。如果没有很多训练数据，这将特别有用。

##config_spacy.yml

```
language: "en_core_web_md"

pipeline: "spacy_sklearn"
```

4.2 训练数据集构建

查找分析股票问询功能主要句式，以及问询的主要问题，设置问询的七个功能有：high price, low price, opening price, closing price, volume, price, change percent.

利用常用句式构建训练数据集 nlu_data.json 如下：


```
{
  "text": "price of TSLA",
  "intent": "price",
  "entities": [
    {
      "start": 9,
      "end": 13,
      "value": "TSLA",
      "entity": "organization"
    }
  ]
},
{
  "text": "What's the volume of WalMart?",
  "intent": "volume",
  "entities": [
    {
      "start": 21,
      "end": 28,
      "value": "WalMart",
      "entity": "organization"
    }
  ]
```


4.3 主体函数构建

调用 iexfinance 中的子函数，构建七个查询子函数。

```
## functional fun
from iexfinance import Stock
def getopen(org):
    return (org.get_open())
def getclose(org):
    return (org.get_close())
def gethigh(org):
    return (org.get_previous()['high'])
def getlow(org):
    return (org.get_previous()['low'])
def getvolume(org):
    return (org.get_volume())
def getprice(org):
    return (org.get_price())
def getchangepercent(org):
    return (org.get_previous()['changePercent'])
```

4.4 Policy

通过提取到的不同意图调用不同子函数。

```python
#policy
from iexfinance import Stock
def policy(message):
    intent = respond(message)[0]
    org = respond(message)[1]
    org = Stock(org)
    if intent == "opening price":
        return  getopen(org)
    if intent == "closing price":
        return  getclose(org)
    if intent == 'high price':
        return  gethigh(org)
    if intent == "low price":
        return  getlow(org)
    if intent == "volume":
        return  getvolume(org)
    if intent == "price":
        return  getprice(org)
    if intent == "change percent":
        return  getchangepercent(org)
```

4.5 Respond

添加简单应答功能，使其更生动。构建回答句式。

```python
#response
def response(message):
    intent = respond(message)[0]
    data = policy(message)
    a = len(intent)
    if intent == "g":
        return "Wish u a happy day:)"
    if a!=1 :
        org = respond(message)[1]
        if intent == "volume":
            return "The {} of {} is {}".format(intent,org,data)
        if intent == "change percent":
            return "The {} of {} is {}%".format(intent,org,data)
        else:
            return "The {} of {} is ${}".format(intent,org,data)
    else:
        return "You can ask me other question:)"
```

4.6 wxpy 集成

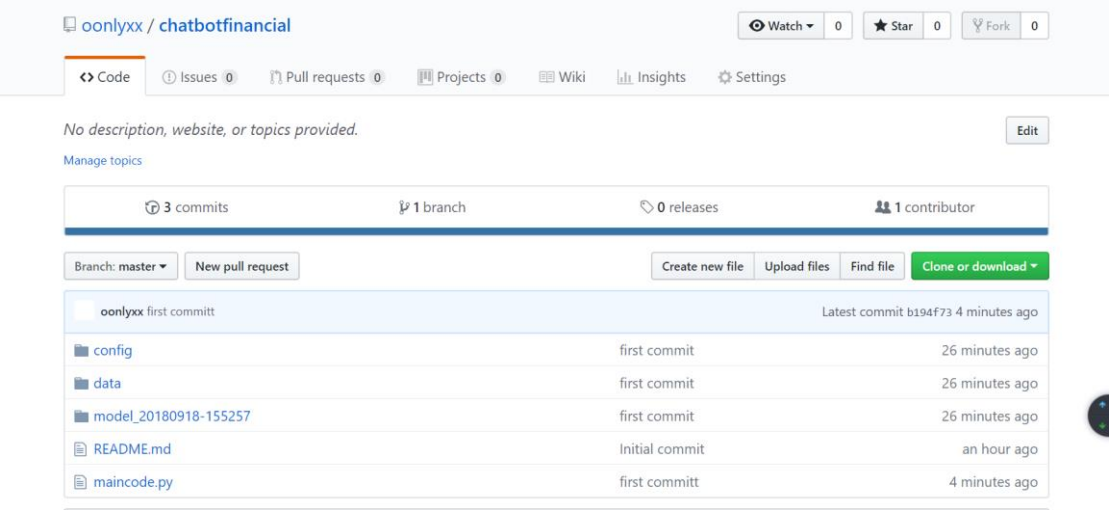
将写好的主函数与微信集成，可实现自我应答或回复朋友的功能。

```python
#wechat log in
from wxpy import *
from wechat_sender import *
bot = Bot()

#回复my_friend发送的消息
my_friend = bot.friends().search(u'your friend name')
@bot.register(my_friend)
def reply_message(msg):
    message = msg.text
    return response(message)
```

4.7 Github

https://github.com/oonlyxx/chatbotfinancial.git



5 问题与解决方案

在实际做大作业时，我也遇到了一些问题：

1. 训练数据时有两种方法：

① `$ python -m rasa_nlu.train -c config/nlu_config.yml -d data/demo-rasa.json -o models/`

② python 调用

这两种方法最终都训练出了数据，但是由于写 data 时粗心多写了一个”:”耽误了一些时间。

2. respond 函数返回多个参数时，返回类型为元组。将其 2 个返回值直接作为参数传入下一个函数则出现问题。尝试多种方法后，选择将元组的值一个个赋值传入，问题解决。
3. 因为 python 输出值有 ‘’ 和 “” 的问题，造成语法错误
4. 还有一些问题就是 python 函数不熟练之类的问题。

总之，在这个过程中也加强了自己对 python 语言的使用熟练度。

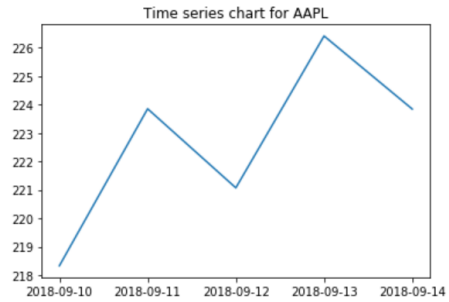
四、 收获与展望

之前一直觉得聊天机器人是一个很神奇的东西，尤其是回答特别人性化的时候。如果有一天自己也能做出来，那真是感觉太不可思议了。在 8 月的四周里，跟着老师和几位同学雷打不动的在周二早上七点半，接触一个自己之前完全不了解的领域，尤其是一些代码写得言简意赅，一些函数之前完全没见过，都对我来说是个不小的挑战。

之前一直不明白老师为什么第一节课花那么长时间去讲背景知识，感觉跟后面的写代码没什么关系。到后来自己开始做大作业时才明白一个大的框架已经建起，这与第一节课老师的熏陶是分不开的。

整个项目还有些不是很完善的地方，比如最开始想做一个可以返回股价波动图的函数，但由于传入参数的调试有问题搁浅了。还有就是中文的查询功能。这些功能在后续会逐步完善。

In [17]:
```
import pandas
from iexfinance import get_historical_data
from datetime import datetime
import matplotlib.pyplot as plt
start = datetime(2018,9,10)
end = datetime(2018,9,15)
f = get_historical_data("AAPL", start, end, output_format='pandas')
plt.plot(f["close"])
plt.title('Time series chart for AAPL')
plt.show()
```



很感谢这次的学习经历，感谢老师课上和课下的耐心指导，感谢和几个同学们一起学习，课后分享讨论。在这个过程中，之前自己写代码很不细心，这次显然也有一些问题，但是也能静下心来一步步 output 检查；也逐渐形成了 ”出现问题-查文档-Google-尝试不同方法-Test-解决”的模式，而不是一味的陷入一个问题，感受到了自己的成长，无论是从代码能力还是从解决问题的能力。