聊天机器人项目报告

吉林大学 车辆工程专业 王一舟

1. 自然语言处理

1.1. 背景

人类的逻辑思维以语言为形式，而采用自然语言与计算机进行通信正是人们长期以来所追求的。实现人机间自然语言通信意味着要使计算机既能理解自然语言文本的意义，也能以自然语言文本来表达给定的意图、思想等。因此，自然语言处理又包括自然语言理解和自然语言生成两个方面。

关于自然语言的理解，其难点便在于人类的语言往往是具有歧义的。人类的话往往会语义隐晦，而这对机器人而言并不是一种容易接受的方式。此时，人们往往会让机器接受大量的训练，从而达到理解人类语言的目的。这就好比让机器人看遍世界上所有的猫，那么当其看到任何一只猫时，都能迅速地辨别出来。这固然是一种让机器人认识“猫”的一种方式，可时过境迁，这世界上所有的猫并不都是一成不变的。要使机器人总是能够保持对猫的辨别，便意味着机器人需要不断地进行大量的训练与学习，而这背后也同时意味着无比庞大的工作量。回到自然语言处理中来，如果能够让机器人识遍所有的人类语言，或者说，领悟到绝大多数常用的语言表达方式，加之机器人适当的推测，便自然能够实现对人类语言的正确理解。

1.2. 实体抽取

1.2.1. RASA

RASA 堆栈是一组开源机器学习工具，供开发人员创建上下文人工智能助手和聊天机器人，其包含 RASA Core 与 RASA NLU 两部分。RASA Core 是基于机器学习的对话管理聊天机器人框架，而 RASA NLU 是意图分类和实体提取的自然语言理解库。

NLU 通过之前的训练数据，便可理解用户的信息，主要是能够实现意图识别和实体抽取两大功能。

1.2.2. 词向量

无论是机器学习还是深度学习，其本质上都是对数字的处理，而词向量（Word Embedding）做的事情就是将单词映射到向量空间里，并用向量来表示。对于相似的词，其

对应的词向量也相近。

深度学习模型具有比传统模型更强的特征抽取能力。在自然语言处理中，传统统计特征包含的信息量过少，这也一直限制着深度学习在自然语言处理中的应用。词向量由于包含了更丰富的信息，使得深度学习能够处理绝大多数自然语言处理应用。

当数据集越大时，结果往往越为准确，但却难以规避一词多义的现象。此外，词向量的训练采用无监督方式，不能很好的利用先验信息。不过，随着对词向量研究的深入，这些问题都将被逐渐解决。

1.2.3. 运用 spacy

Spacy 是一个高级的 NLP 库，其包含了一个快速的实体识别模型，它可以识别出文档中的实体短语。Spacy 可识别多种类型的实体，例如人物，地点，组织，日期，数字。Spacy 使用的语言模型是预先训练的统计模型，能够预测语言特征，对于英语，共有 en_core_web_sm、en_core_web_md 和 en_core_web_lg 三种语言模型，使用 spacy.load()函数来加载语言模型。语言模型中不仅预先定义了 Language 管道，还定义了处理文本数据的处理管道（pipeline）。

使用 spacy 时，首先需要引入 spacy 库：

```
import spacy
```

本项目采用了 en_core_web_md 语言模型：

```
nlp = spacy.load("en_core_web_md")
```

通过定义 include_entities 来存放需要抽取的实体类型。例如，“DATE”表示日期，“ORG”表示组织，“PERSON”表示人物。

```
include_entities = ['DATE', 'ORG', 'PERSON']
```

通过定义函数实现实体的提取：

```
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:
            # Save interesting entities
            ents[ent.label_] = ent.text
    return ents
```

例如，当输入“Jack graduated from MIT in 2016”时，机器人将自动从中抽取“Jack”“MIT”“2016”三者，并分别将其归类至“PERSON”“ORG”“DATE”当中，从而有效地实现实体的抽取。

1.3. 正则表达式

正则表达式描述了一种字符串匹配的模式，可以用来检查一个串是否含有某种子串、将匹配的子串替换或者从某个串中取出符合某个条件的子串等。正则表达式的灵活性、逻辑性和功能性非常强，并且可以迅速地用极简单的方式达到字符串的复杂控制。

本项目主要将正则表达式应用到闲聊的语境当中，针对常用的三种用法：“thank”表示感谢，“what can”表示提问，“bye”表示道别（同时也是终止该运行程序的一种标志），设计了以下的 rules：

```
rules = {'thank(.*)': ["My pleasure"],
         'what can (.*)': ["I'm a robot to help you get information about stock"],
         'bye(.*)': ['Ok, bye']
        }
```

字典 rules 中的每个 key 作为 pattern，而所对应的 value 则作为 response 输出：

```
def chitchat_response(message):
    # Call match_rule()
    response, phrase = match_rule(rules, message)
    # Return none is response is "default"
    if response == "default":
        return None
    if '{0}' in response:
        response = response.format(phrase)
    return response

def match_rule(rules, message):
    for pattern, responses in rules.items():
        match = re.search(pattern, message)
        if match is not None:
            response = random.choice(responses)
            var = match.group(1) if '{0}' in response else None
            return response, var
    return "default", None
```

2. API 查询

2.1. iexfinance

本项目旨在查询股票信息，而 API 则是一个获得股票信息的有效渠道。通过调用 iexfinance 中 stocks 的数据，便可以实时获得股票信息。

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

本项目可实现对特定公司股票的特定信息类型的查询。

2.2. 公司名称识别

为了提升程序的鲁棒性，在识别公司名称时，考虑到人们不同的输入习惯以及输入出错的可能性，程序中列举出了多种可能结果。以苹果公司为例，stocks 中苹果公司的简称为 AAPL，但人们在查询苹果公司的股票时，并不一定知道这一简称。因此，当人们输入“apple”时，也能够有效地将其识别出来。此外，即使人们误输入为“aple”，也依然可以查询到苹果公司的股票。

```
def company_identification(message):
    if 'aapl' in message or 'appl' in message or 'aple' in message:
        return 'AAPL'
```

以上只是苹果公司的查询方式，本项目还可实现对 Google, BlackBerry, HP, IBM, Nokia, Microsoft, Baidu, Alibaba, Sina, SOHU, NetEase 公司的查询。由于这些公司名称的识别方式与苹果公司类似，因此代码不再给出。

2.3. 查询类型识别

为了丰富股票查询的内容，本项目设计了对多种股票信息类型的查询，包括 latest price, lowest price, highest price, volume, logo, historical information。

```
def item_idetification(message):
    if 'low' in message:
        return 'low'
    if 'high' in message:
        return 'high'
    if 'price' in message:
        return 'latestPrice'
    if 'volume' in message:
        return 'volume'
    if 'logo' in message:
        return 'logo'
    if 'historical' in message:
        return 'historical'
    return None
```

2.4. 股票信息查询

通过下列代码中的 get_quote() 可获得 latest price, lowest price, highest price, volume 等信息。此外，若将 get_quote()更改为 get_logo()，即可查询到公司的 logo 信息。

```
info = Stock(comp,token="pk_910d4c5b9a2d4d379af7260ae8e549f2")
item = item_idetification(message)
API_search = info.get_quote()
```

当获取 historical information 时，自然会涉及到日期的问题。本项目通过 spacy 作实体

抽取，从中提取“DATE”作为股票历史信息查询的日期。由于周末股市休市，因此需对抽取的日期先进行判断：

```
if datetime.strptime(date,'%Y %m %d').weekday() == 0 or datetime.strptime(date,'%Y %m %d').weekday() == 6:
    response = "Sorry. I can only find stock information on weekdays."
else:
```

若输入日期当天确定为工作日，则采用 get_historical_data 来获得当天的股票信息。

```
start = datetime(2015, 1, 1)
end = datetime(2019, 8, 14)
API_search = get_historical_data('AAPL', start, end, output_format='pandas',token="pk_
date_dict = API_search.loc[date]
```

3. 状态机

3.1. 概述

在机器人的问询过程中，必然会产生多轮问询的状况。此时，上一轮问询的结果必须要被保存下来，并在该轮问询中进行调用。因此，引入了状态机的概念。

状态机实际上就是一个状态的转移图。举个最简单的例子：人有三个状态：健康、感冒、康复中。触发的条件有淋雨(t1)、吃药(t2)、打针(t3)、休息(t4)。于是人的状态变化便存在以下情况：健康->(t4)->健康；健康->(t1)->感冒；感冒->(t3)->健康；感冒->(t2)->康复中；康复中->(t4)->健康等等。状态机就是指某一状态在不同的条件下，跳转到自己或其他状态的图。因此，状态机可归纳为 4 个要素，即现态、条件、动作、次态。“现态”和“条件”是因，“动作”和“次态”是果。

3.2. 实际应用

本次项目中定义了六种状态，以实现对股票信息的多轮问询：

```
INIT=0
AUTHED=1
CHOOSE_COMPANY=2
CHOOSE_ITEM=3
CHOOSE_DATE=4
DATE_DONE=5
```

“INIT”状态为初始状态，经过登录（输入数字）后，便可进入“AUTHED”状态，此时便可以输入所需要查询的股票公司了，并由此进入“CHOOSE_COMPANY”状态。选择完股票公司后，便需要输入所需查询的项目类型，并由此进入“CHOOSE_ITEM”状态。由于查询 historical information 时需要额外输入日期，因此进入“CHOOSE_DATE”状态。当输入完日期后，便进入“DATE_DONE”状态。为此，特意设计了如下的 policy_rules 字典：

```
policy_rules = {
    (INIT, "search"): (INIT, "You'll have to log in first, what's your phone number?", AUTHED),
    (INIT, "number"): (AUTHED, "Perfect, welcome back!", None),
    (AUTHED, "search"): (CHOOSE_COMPANY, "Which company would you like to know about?", None),
    (CHOOSE_COMPANY, "specify_company"): (CHOOSE_ITEM, "What kind of stock information for {0} would you like to see?", None),
#choose item, the second is to back to choose company
    (CHOOSE_ITEM, "specify_item"): (CHOOSE_ITEM, "Ok! The {0} is {1}", None),
    (CHOOSE_ITEM, "specify_company"): (CHOOSE_ITEM, "What kind of stock information for {0} would you like to see?", None),
#choose historical, the second is to back to choose item, the third is to back to choose company
    (CHOOSE_DATE, "historical"): (DATE_DONE, "Ok! But you have to tell me the date first.", None),
    (CHOOSE_DATE, "specify_item"): (CHOOSE_ITEM, "Ok! The {0} is {1}", None),
    (CHOOSE_DATE, "specify_company"): (CHOOSE_ITEM, "What kind of stock information for {0} would you like to see?", None),
#choose date, the second is to back to choose historical, the third is to back to choose item, the fourth is to back to choose company
    (DATE_DONE, "date"): (DATE_DONE, "Ok! Here you are: {0}", None),
    (DATE_DONE, "historical"): (DATE_DONE, "Ok! But you have to tell me the date first.", None),
    (DATE_DONE, "specify_item"): (CHOOSE_ITEM, "Ok! The {0} is {1}", None),
    (DATE_DONE, "specify_company"): (CHOOSE_ITEM, "What kind of stock information for {0} would you like to see?", None)
}
```

此外，在处于偏后的状态时，都需要可以返回到前面任意一个状态，这样才能够保障股票查询的灵活性与多样性。因此，在设计上述 policy_rules 时，需要将各个不同的状态相联系。

4. 总结

通过本次项目，我提升了自己的 python 编程能力，更重要的是，体验到了参加科研并完成一个完整项目的过程及其所需付出的艰辛，学会了如何应对各种各样的计算机学习中的问题，而这些对我的能力提升帮助很大。

在参加这一项目之前，我只学习过 C 语言，还远谈不上是一个会编程的人。因此，最开始的 python 语言学习，对我便是一个不小的挑战。所幸经过我初期对一些基础知识的自学，加之在课堂上老师对一些重要 python 知识的讲解，python 我还是比较快地完成了入门。在之后的课程当中，随着学习的深入，我也有了更深一步的思考，也尝试了自己编一些 python 程序以作练习。现在回想来，能够以这样一种方式走进 python 学习的大门并进行了一定的编程实践，也着实是一件令我感到很欣慰很值得的时事情。

几次公开课程授课结束，便进入了最后的大作业环节。起初我并不是很有信心能够自主完成整个项目，为此我还特意把老师每一次课讲的关键内容以及老师编的每一行代码都温习了一遍。还记得，编程过程中遇到的最麻烦的问题，便是状态机的设定。或许是我自己对状态机部分理解的还不够深入，抑或是状态机本就比较晦涩，加之我在代码中引入的状态数较

多，便使得程序比较繁琐。

总之，我相信程序一定还有不少的改进空间，也希望自己日后能够随着学习的深入将其完善。