人工智能项目学习报告

——聊天机器人

学校：华中科技大学

姓名：叶壮博

邮箱：zb_ye@foxmail.com

日期：2019 年 5 月 26 日

目　录

1. 研究背景

1.1 自然语言处理

自然语言处理并不是一门新兴的学科，从 50 年代的机器翻译和人工智能研究算起，NLP（Natural Language Processing，自然语言处理)已有长达半个世纪的历史。

自然语言处理包括多个方面和步骤，其完成的基本功能有认知、理解、生成等等。自然语言处理需要理解从外部输入的语言，对其进行相应的处理。因为现实世界是含糊的，充满着各种不确定因素，而语言经过千百往年的发展，有着极其丰富的内容，这给自然语言处理带来了很大的困难。如何消除词的歧义？如何理解模糊的句子？如何应对有瑕疵或者不规范的输入？如何理解语言与行为的关系？这里的每一个问题，都给自然语言处理的研究带来了阻碍。

随着自然语言处理的发展，不断有新的技术被应用到了这一领域。语料库、语料库语言学等技术的加入，使得大规模真实文本的机器学习处理成为自然语言处理的主要选择。统计数学方法越来越受到重视，自然语言处理中越来越多地使用机器自动学习的方法来获取语言知识。

1.2 聊天机器人

聊天机器人是经由对话或文字进行交谈的程序。聊天机器人能够模拟人类对话，一般用于实用的目的，通常完成的工作为客户服务或者咨询获取。

与自然语言处理类似，聊天机器人的出现也非常早。由系统工程师 Joseph Weizenbaum 在 20 世纪 60 年代共同编写的 ELIZA 是世界上第一个真正意义的聊天机器人。ELIZA 会对传入的语句中进行关键词扫描，为其中的某些“关键词”匹配上合适的“对应词”，然后再将处理后的结果输出。

聊天机器人使用到的人工智能领域就是我们之前提到的自然语言处理。从最开始的 ELIZA 到现在，自然语言处理的发展使得聊天机器人变得愈加“智能”，愈加实用。

聊天机器人通常被整合到对话系统中，以虚拟助手的形式活跃于各个即时通信平台。这些与娱乐、客服、导购为目的的聊天机器人为我们的生活带来了极大的便利。

2. 项目要求

1) 实现一个用于查询足球数据的聊天机器人，要求可以查询五大联赛的比赛、

排行榜和当前正在进行的比赛。

2) 将聊天机器人整合到微信中，可以通过微信与聊天机器人交流并获得足球数据。

3) 让聊天机器人具有基本的聊天功能，可以回答一些基本的句式，并对类似于打招呼的语句进行回复。

3. 项目设计与实现

3.1 微信接口

使用 wxpy 库来完成相应的功能，具体实现步骤如下：

1) 使用 Bot()新建机器人对象，运行时会弹出一张二维码图片。通过微信客户端扫描二维码即可用客户端的账号登录微信。

2) 通过 Bot.friends().search("friends name")函数找到特定微信好友，再通过 Bot.register(friend)作为函数的装饰器，重写 forward_message(msg)函数就可以读取该好友传过来的消息了。

3) 在 forward_message(msg)函数中调用消息处理函数，将处理结果返回，即可将处理后的话发送给该微信好友

4) 通过 Bot.join()函数堵塞线程，使得该线程保持监听状态。

该部分实现代码如下所示：

```
from nlu_process import main
from wxpy import *

bot = Bot()
my_friend = bot.friends().search('Chat_Bot')[0]
res = main.Respond()


@bot.register(my_friend)
def forward_message(msg):
    if msg.type == 'Text':
        print("USER: {}".format(msg.text))
        response = res.send_message(msg.text)
        print("BOT: {}".format(response))
        return response
    else:
        return "please use text to chat with me"

bot.join()
```

3.2 常用句式的识别和回应

该部分需要识别常用句式，并进行相应回复。这有些类似 ELIZA 实现的功能。

1) 构建一个名为 rules 的字典，其中 keys 部分为匹配的模板，values 部分为该模板回复的句型。keys 部分的模板以正则表达式的方式编写，如 key 为“do you think (.*)”，value 为一个 list，list 中每个值都能作为该模板的回复，如['if {0}? Absolutely.', 'No chance']。
2) 构建一个 match_rule 函数对模板进行匹配。对于 rules 中的每一个 key，都通过 re.search 对 key 和传入的消息进行匹配。匹配过程中使用 re.IGNORECASE 忽略大小写。如果匹配成功，则将传入消息中除了模板部分的话和 value 返回，否则返回 None
3) 这样当我们调用 match_rule 检测到匹配成功时，我们就可以将 value 与原本句子中的话组合在一起，作为消息返回。

该部分实现代码如下所示：

```
rules = {'if (.*)': ["Do you really think it's likely that {0}", 'Do you wish that {0}', 'What do you think about {0}', 'Really--if {0}'],
'do you think (.*)': ['if {0}? Absolutely.', 'No chance'],
'I want (.*)': ['What would it mean if you got {0}', 'Why do you want {0}', "What's stopping you from getting {0}"],
'do you remember (.*)': ['Did you think I would forget {0}', "Why haven't you been able to forget {0}", 'What about {0}', 'Yes .. and?']}

# Define chitchat_response()
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:
        # Replace the pronouns of phrase
        phrase = replace_pronouns(phrase)
        # Calculate the response
        response = response.format(phrase)
    return response

```

```
def match_rule(rules, message):
    for pattern, responses in rules.items():
        match = re.search(pattern, message, re.IGNORECASE)
        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

def replace_pronouns(message):

    message = message.lower()
    if 'me' in message:
        return re.sub('me', 'you', message)
    if 'I' in message:
        return re.sub('I', 'you', message)
    elif 'my' in message:
        return re.sub('my', 'your', message)
    elif 'your' in message:
        return re.sub('your', 'my', message)
    elif 'you' in message:
        return re.sub('you', 'me', message)
```

3.3 意图和实体识别方法

1. 对使用到的技术的介绍
   1) spaCy 是一个 Python 自然语言处理工具包，包含多种用于自然语言处理的工具。en_core_web_md 是 spaCy 使用的三种语言模型中的一种，用于预测语言的特征。在 en_core_web_md 中包含长度为 300 的词向量数据，我们可以通过这些词向量分析两个词之间的相似程度等信息。

      对于意图识别，我们可以利用 spaCy 和分类器来完成。常用的分类器有 scikit-learn 中的最近邻分类器和支持向量机。

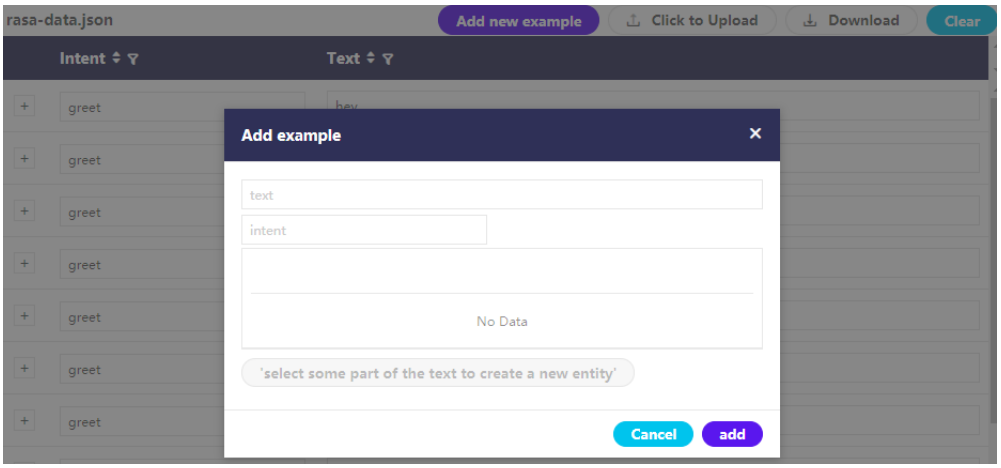
      此外还能运用 spaCy 进行实体抽取，提取一句话中的实体和实体之间的依赖关系。
   2) Rasa 是一个用于构建语言相关的 AI 助手和聊天机器人的机器学习开源框架。Rasa 有两个主要模块，NLU 模块用于理解用户输入的消息，Core 模块用于保持聊天并且决定应该做什么。Rasa 是基于 spaCy、scikit-learn 来实现的。因此我们上文提到的 spaCy 实现的功能，在 Rasa 中得到了整合。Rasa 一般用来做意图识别和实体提取的工作。
2. 运用 Rasa 进行本项目的意图与实体识别
   1) 构建用于识别的数据

Rasa 用于识别的数据是 json 格式的。每一条数据有由 test、intent 和 entities 三部分构成。test 代表着用于识别的一句话，intent 为这句话想要表达的意图，entities 为这句话中包含的实体。一条典型的 rasa 训练数据如下所示：

```
{
  "text": "which team ranks first in the La Liga",
  "intent": "rank_search",
  "entities": [
    {
      "start": 30,
      "end": 37,
      "value": "La Liga",
      "entity": "competition"
    }
  ]
},
```

可以看到在 entities 项中我们表明的实体的开始和结束位置，实体的值和所属的类型。

依照这样的思路，我们需要对本聊天机器人可能需要识别的意图编写对话，标出对话的意图和实体，以上述格式整理成 json 文件。我们可以通过 https://rasahq.github.io/rasa-nlu-trainer/ 这个网站来完成对 json 文件的在线编写。如下图所示，在弹出的窗口中填写对应的 text、intent、entities 后，网站自动将所有数据整理成 json 文件。



2) 通过训练数据训练 interpreter

最终我们可以识别的 intent 有："rank_search"，"match_search"，"live_match_search"，"country"，"greet"，"goodbye" 这 6 个。在 config_spacy.yml 文件中，我们设置选择的库为 en_core_web_md，使用的 pipeline 为 spacy_sklearn，之后就可以进行 intepreter 的构建了。

通过 rasa_nlu.config.load 读取 config_spacy.yml 文件并通过 Trainer

函数构建训练器对象。通过 load_data 读取我们之前生成的 json 文件后，通过 Trainer.train 函数就可以对训练数据进行训练并返回我们最终需要的 interpreter 了。

在之后的检测过程中，通过 data = interpreter.parse(message)分析输入的 message，data 中是分析后的意图和实体。

3.4 对问好与道别的处理

1) 对于问好的处理。

当识别意图为"greet"的时候，代表传入的消息是在问好。而问好的时候一般会加入自己的名字，例如“hello! my name is Mike”这类信息。因此我们在对问好的处理时希望把姓名提取出来对对方进行回应，例如回复“hi, Mike. it’s nice to meet you”，这样显得聊天对话更加自然。

最开始的时候采用正则表达式的方式进行识别，识别所有首字母大写的单词，将这些单词组合成姓名。但是这样的方式很容易遇到问题，例如发过来的消息首字母大写或者使用小写字母拼写自己的姓名，都会导致姓名识别的失败。

后来我希望在识别意图为"greet"的时候，加入对姓名的实体判断。也就是在 rasa_nlu 中完成对姓名的提取。但是这样操作也遇到了一些问题，对于在训练数据中没有出现过的姓名，rasa_nlu 的识别效果并不好。而世界上不同的英文名实在太多，将所有这些都作为训练数据显然不现实。

最后我选择通过 spaCy 中的识别方式完成这个功能。spaCy 的 en_core_web_md 库中包含了很多不同的姓名，通过这些自带的数据进行识别，可以在更大范围内实现姓名识别，具体的实现方法如下：

```
1. import spacy
2. nlp = spacy.load("en_core_web_md")
3.
4.
5. # Define included entities
6. include_entities = ['DATE', 'ORG', 'PERSON']
7.
8.
9. # Define extract_entities()
10. 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
```

在主函数中调用该功能的实现：

```
if intent["name"] == "greet":
    tmp = et.extract_entities(message)
    if tmp["PERSON"] is None:
        results = random.choice(nl.greet_responses1)
    else:
        results = random.choice(nl.greet_responses2).format(tmp["PERSON"])
    return results, params, next_state
```

2) 对于回复多样性的处理

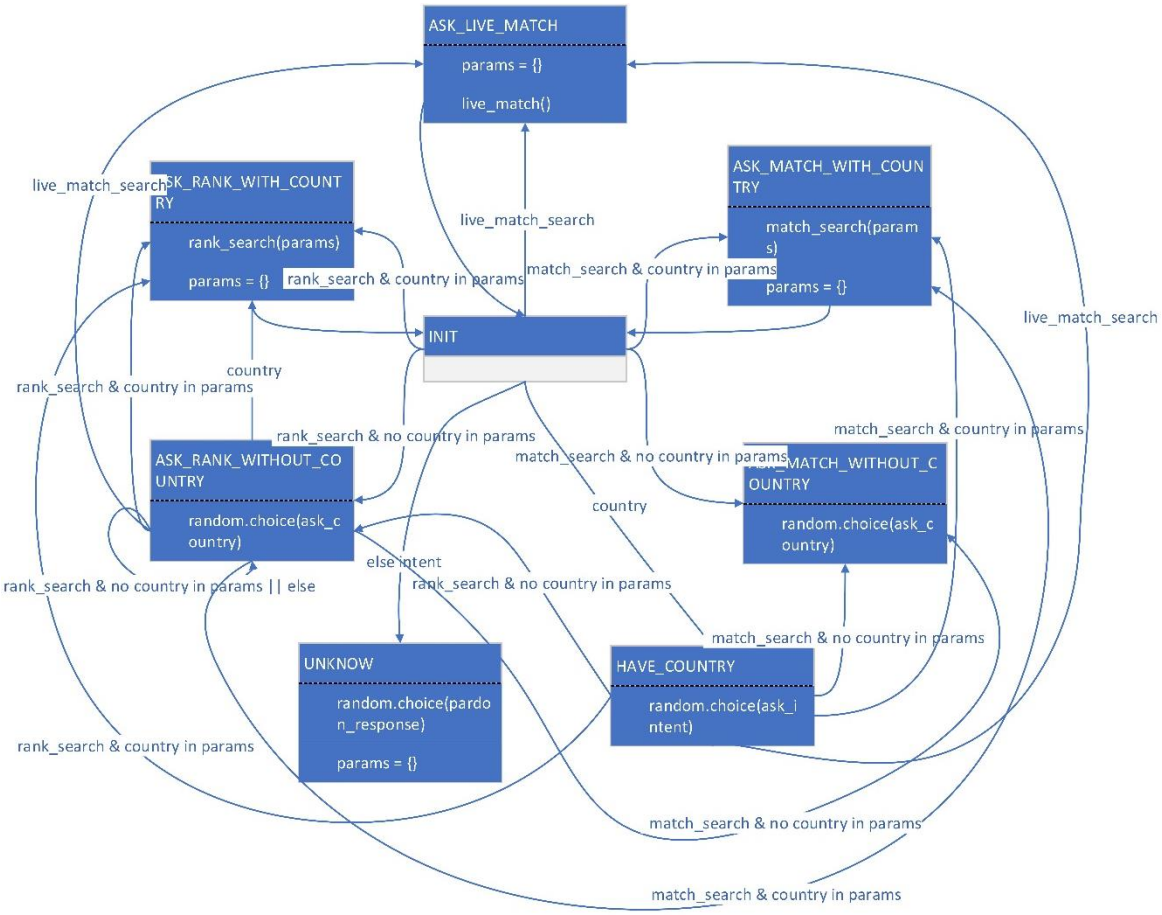
在进行回复时，就算我们已经确定了对方的意图以及回复的内容，我们也可以让回复更具有多样性，具体的实现方法就如上图所示，通过random.choice 在实现写好的回复列表中选择语句进行回复。

3.5 状态转换的设计与实现

为了让聊天机器人可以应对现实对话中的“用多个短句来对想法进行补充”和“对话具有状态”的特性，我们为聊天机器人设置状态机。

1) 状态机的设计

根据可以识别的意图，我们设计如下 8 种状态：INIT、ASK_RANK_WITH_COUNTRY、ASK_RANK_WITHOUT_COUNTRY、ASK_MATCH_WITH_COUNTRY 、 ASK_LIVE_MATCH 、 ASK_MATCH_WITHOUT_COUNTRY、HAVE_COUNTRY、UNKNOW。状态转移图如下所示：



为了不使图片看起来太过复杂，在图片中省略了 ASK_MATCH_WITHOUT_COUNTRY 部分的状态转移，因为这部分可以类比 ASK_RANK_WITHOUT_COUNTRY。

2) 根据此状态转移图，实现状态转移函数，实现结果如下：（仅展示部分结果，用作示例）

```
# Define the states
INIT = 0
ASK_RANK_WITH_COUNTRY = 1
ASK_RANK_WITHOUT_COUNTRY = 2
ASK_MATCH_WITH_COUNTRY = 3
ASK_MATCH_WITHOUT_COUNTRY = 4
ASK_LIVE_MATCH = 5
HAVE_COUNTRY = 6
UNKNOW = 7


# change the state
def porcedure(state, intent, params):
```

```
    if state == INIT:
        if intent == "rank_search":
            if "competition" in params.keys():
                return ASK_RANK_WITH_COUNTRY
            else:
                return ASK_RANK_WITHOUT_COUNTRY
        elif intent == "match_search":
            if "competition" in params.keys():
                return ASK_MATCH_WITH_COUNTRY
            else:
                return ASK_MATCH_WITHOUT_COUNTRY
        elif intent == "live_match_search":
            return ASK_LIVE_MATCH
        elif intent == "country":
            return HAVE_COUNTRY
        else:
            return UNKNOW
```

3.6 足球数据接口

接下来就是最后一部分，足球数据接口的实现了。选择的足球 api 为 https://www.football-data.org/，接口相关代码参考其官方 cli，网址为 https://github.com/architv/soccer-cli，为了适应本聊天机器人的功能需求，对部分代码做了修改。

需要专门说明的是，预期做的功能里面包括了查询正在进行的比赛，通过我的状态机设计和代码都可以看到有该功能的相关代码。但是可能是因为各大足球联赛相继结束，该 api 网站不再提供该查询功能，导致我在项目进行过程中发现原本可以调用的函数总是显示连接超时而无法使用。

为了解决这个问题，我尝试和该 api 提供方进行邮件联系，但截至我开始着手书写项目报告的时候，该网站仍未回复我的邮件。迫不得已我只能在最终的展示视中删除这一功能。

4. 感想与总结

1. 总结

在本次的人工智能项目中，在老师的带领下，我们从最简单的回声聊天机器人开始，学习了这个领域的很多知识。

在第一次课中，我们学习如何用正则表达式处理文本，预先设置好模板和相应回答，通过正则表达式将收到的对话与模板进行匹配，再将该模板对应的回答与传入的对话中的另一部分结合，作为最终的输出。

在第二次课中，我们的重点是学习如何识别一句话中的意图和实体。最开始我们仍通过正则表达式，通过识别特定的关键词可以识别意图，通过识别那些首字母大写的词可以识别实体。当然，这样的方法显然不能让人满意。接着我们学习了词向量来表示单词的含义，了解了 spaCy 和 en_core_web_md 等库，以及用向量角度来表达词与词之间的相似度。在有了这些基础之后，我们学习了 scikit-learn 中的最近邻分类和支持向量机。之后我们运用 spaCy 进行了实体抽取和实体间依赖分析的实践。

第三次课我们介绍了 rasa 这个工具，以及如何通过 rasa 来进行意图与实体识别。在有了这个强有力的工具后，我们开始利用 rasa 来进行语言分析。我们以酒店和餐厅预定为例子，通过 rasa 分析出意图以及实体，通过意图选择执行功能，再将实体用作参数执行个性化查找。在课堂上，我们与数据库进行交互，查找数据库中符合要求的餐厅或者酒店。之后我们学习了增量槽填充和否定的相关知识，通过这些技术可以让聊天机器人具有“记忆”，用户可以通过多个较短的命令让聊天机器人执行特定操作。

第四次课我们主要学习如何在聊天机器人里构建状态机，并实践了如何实现询问队列和问题队列，所有这些都是让我们的聊天机器人的交流功能更加符合人的日常对话习惯。在这之后，我们将这所有 4 次课的知识整合，实现了一个建议的咖啡预定聊天机器人。

2. 感想

通过这段时间的学习，我对 NLU 和聊天机器人有了相比之前更加深刻的理解。在学习过程中，我了解了聊天机器人中会用到的各种技术，不同技术间的差异以及他们解决的问题。在课堂上我通过代码学习如何使用这些技术，这也加深了我对他们的理解。

在最后的项目完成部分，则是运用上课内容进行实践。实践的过程总会让我们遇到新的问题和挑战，这些往往是我们在学习理论知识过程中容易忽略的，因此项目过程中，我又对课上学习的知识有了新的理解。为了使得自己的代码对语句有更好的分析效果，我尝试不同的方法对其进行修改优化，并最终达到了让我满意的效果。我认为这个过程是必要且令我受益匪浅的。

总的来说，本次项目让我收获颇多，非常感谢老师与另外两位同学的陪伴。

5. 参考资料

1) https://github.com/architv/soccer-cli
2) 课程 PPT、课程学习代码