智能聊天机器人
项目总结报告

项目：智能聊天机器人

学校：东南大学

专业：计算机科学与技术

姓名：杨航源

项目日期：2019.8.6-2019.9.22

目录

研究背景..... 2

概念与应用..... 2

发展前景 3

知识点回顾及应用 4

项目总结..... 11

参考资料..... 11

一、 研究背景

1. 概念与应用

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，通过图灵测试，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。

目前聊天机器人广泛运用于即时通讯平台，即时通讯平台提供易于整合的webhook，使得第三方开发商易于可通用于不同通讯平台之聊天机器人。这些软件机器人以客服的身份出现或是成为团体聊天的一员。有些即时通讯的机器人可以连接外部数据库，提供使用者新闻，气象，导航，电影放映时间，股价等资讯。达美乐、必胜客、迪士尼、Nerdify、雅玛多 Line、全食超市都已推出各自的聊天机器人，以便与终端消费者增进交流，推销公司的产品与服务，并且让消费者订货更加方便。2016 年，观光业的一些旅行社和航空公司透过 Messenger 推出了聊天机器人的服务，墨西哥航空利用人工智能售票、回答问题，中国的旅行社在此之前已用 Wechat 提供这些服务。

有些聊天机器人，例如 Nerdify 开发的 Nerdy Bot，针对大中小学生面对的问题，让学习更简单又有效率。该软件利用脸书 Messenger 即时回答学生作业相关的问题以便加速学习。加大尔湾分校图书馆的聊天机器人 ANTswers，2014 年开始试用，被认为非常成功。

2. 发展前景

云平台相关开发工具的投资增多，为了让更多的用户实现通过更自然的语言和文本与应用程序进行交互，许多开发人员开始转向公有云来进行创建，在科技巨头中，如亚马逊、微软和谷歌也都在其云平台中投资了大量聊天机器人开发工具，使得开发人员可以更轻松的为面向用户和企业的应用程序创建对话界面。

亚马逊：AWS Lex 是一项包含自然语言理解功能的服务，它基于亚马逊的核心技术 Alexa 之上。开发人员创建的技能本质上是应用程序组件，之后可以将这些技能组合在一起，以构建更复杂的聊天机器人界面。目前，Lex 支持各种各样的用例， 面向消费者的 Lex 应用程序可以运行在超过 1000 万个 Alexa 设备上。同时，亚马逊还将 Lex 定位于企业用途，并为 iOS 和 Android 设备均提供 Lex 软件开发套件，以及 Java，JavaScript，Python，.NET，Ruby on Rails，PHP，Go 和 C ++等网络应用程序。

微软：Bot Framework 构建于 Microsoft Azure，该公司的认知服务库和 Cortana 应用程序。开发人员可以在私有服务器或 Azure Bot 服务中部署聊天机器人。为了获得更好的可扩展性，其机器人还可以在 Azure Functions 上运行。 另外，机器人可以通过各种渠道与用户互动，开发人员可以使用 C＃或 Node.js 进行构建。而微软也可以容易在其机器人库中发布新的机器人，其中包括为特定功能调用第三方机器人服务的工具，比如预定出租车或查询企业的应用程序。

谷歌：Dialogflow 是 google 收购的 API.ai 平台的品牌重塑，开发人员可以使用各种语言构建应用程序，包括 c++，Python，.NET，Ruby 和 Java。另外，开发工具也适用于 iOS，Android，Cordova，Xamarin 和 HTML。之后，谷歌

将继续把核心Dialogflow功能集成到其针对硬件和Android设备的谷歌智能助理服务中，用于分析的 Chatbase 平台及其各种消息服务中。这构建基于Dialogflow 对各种消息传递格式和平台的支持上，包括 Alexa 以及主要的云聊天机器人工具。

二、 知识点回顾及应用

1. 同一个问题多种选择性的回答，并提供缺省回答的方案

定义 response 变量为 python 中的字典，设置 key 为可能出现的 message，value 设置为 list，通过 python 中的 random 库函数 random.choice()来从多种选择性的回答中随机选择，额外设置’default’对应键值对来处理缺省情况。

2. 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

定义 rules 变量为 python 中的字典，key 设置为正则表达式匹配模式，value 提供多种选择性的回答，message 通过 python 中的 re 库函数 re.compile()预编译正则模式并用该函数返回值的 search 方法判断是否存在相应匹配项，通过 findall 函数返回匹配结果，句法转换主要涉及人称的转换，通过一一判断”i”，” you”等是否存在于 message 中再一一置换即可。

项目中的应用于 greet 意图的回答和聊天中意料之外的 message 的处理：intent 为 greet 时调用 find_name()，利用'[A-Z]{1}[a-z]*'的模式提取姓名，用于 greet 中的回答；bot 的 handler 中设置 clean 用于在处理用户 message 之前对于匹配'if (.*)|do you think (.*)|do you remember (.*)')模式的 message 先进行过滤处理，由预先设置的 rules 来选择回复的 message，并通过

replace_pronouns()函数来替换人称；当 message 的 intent 无法被识别时，即 intent 为 None，则默认 intent 为 ask_explanation，回复用于和 bot 交流的提示信息。

3. 通过正则表达式、最近邻分类法等多种方案来提取用户意图

对于正则表达式，可以设置不同意图对应的模式，例如可以分别对于 greet/thank/goodbye 的意图设置匹配模式，并给定多种选择性回答；最近邻分类法、SVM 等分类方法对意图进行分类是基于词向量的概念，利用句子中的词向量获得整个句子对应的向量，并给定已知样本的意图进行训练，从而获得测试样本的意图。

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

预建的命名实体类型主要利用 spacy 中的 entity recogniser，nlp 设置为 spacy.load("en_core_web_md")(也 可 取 其 他 数 据 集 ）， doc 赋 值 为 nlp(message)，即可从 doc.ents 中获得预建的实体，角色关系例如 from A to B 的关系可以通过正则表达式”from（.*）to（.*)”进行匹配从而获得不同匹配 group 之间的关系，也可以通过 spacy 处理结果 doc 中 word 的 ancestors 来获得实体之间的依赖关系，从而确定 from A to B 中 A 和 B 之间的关系或者是 a blue hat and a red coat 中物品和颜色的关系。

5. 基于 Rasa NLU 的本地基础聊天机器人系统的构建

Rasa nlu 的安装:

$ pip install rasa[spacy]

$ python -m spacy download en_core_web_md

$ python -m spacy link en_core_web_md en

代码模板如下：

```
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
trainer = Trainer(config.load("config_spacy.yml"))
training_data = load_data('demo-rasa.json')
interpreter = trainer.train(training_data)
```

之后即可通过 interpreter.parse(message)[“entities”]获得实体识别的结果信息，interpreter.parse(message)[“intent”]获得 message 意图信息。

config_spacy.yml 文件中包含了 spacy 的数据集(pipeline 设置为 spacy_sklearn 时)和 pipeline，用于选择训练方法，例如:

Language: “en”

Pipeline: “spacy_sklearn”

demo-rasa.json 文件用于存放训练数据，可以使用 markdown 格式或者 json 格式，详细格式见 rasa 相关文档。

项目中应用: config_spacy.xml 文件依旧使用 en+spacy_sklearn 的方法，训练文件为：rasa-stock.json，其中包含 regex_features/entity_synonyms/common_examples/lookup_tables 部分，本项目中后两者较为重要。lookup_tables 用于存放预标注的实体，便于识别；common_examples 最为重要，对于给定句式预先设置 intent 和其中的 entity 用于训练，特别要注意经过实验测试每一种句式需要至少出现 5-6 次(仅

entity 不同)rasa_nlu 的训练结果才能较为理想地区分其 intent 以及其中的实体(tips: rasa_nlu 识别出的 entity 并不区分大小写，需要额外注意)

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）

借助于 python 的 sqlite3 进行轻量级的数据库处理，基本方法例如:

```
query = 'SELECT * FROM hotels'
//可以用如下方式连接条件
if len(params) > 0:
    filters = ["{}=?".format(k) for k in params]
    query += " WHERE " + " AND ".join(filters)
t = tuple(params.values())  //t 为 value 的元组用于查询，预防 SQL 注入
conn = sqlite3.connect("hotels.db")      //连接数据库
c = conn.cursor()
c.execute(query,t)    //执行 SQL 语句
print(c.fetchall())    //获取查询结果
```

项目中运用了类似的数据库查询语句，并添加的 neg_params 用于处理否定条件， 同时处理了大小写的细节问题，例如将 filters = ["{}=?".format(k) for k in params]改写为 filters = ["LOWER({})=?".format(k) for k in params]并预先将 params 全部改为小写，从而避免由大小写出现的问题。

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

基于增量过滤器的单轮多次增量查询技术是建立在单次查询的基础之上的，设置 params, neg_params 变量用于存储对于不同实体的选择/否定，respond 函数在返回 response 需要的 message 的同时返回通过本次 message 获得更

新的新的 params, neg_params，从而更新参数。

对于否定实体的甄别，仅仅考虑实体所属句段是否存在 n’ t 以及 not 用于鉴别是否为否定实体，具体方法为借助于识别出的实体对句段进行分割，在不同句段中进行判断。

项目中应用：用于 confine 意图的处理(在选择 stock 且未 affirm 之前)，在对状态机进行状态转换后若 response 为 None 则说明需要进行 stock 的查询，通过 respond 方法获取查询结果的同时更新 params 和 neg_params。(tips: 若对于同一类型的实体在多次查询条件中出现，则仅仅在 paramshe neg_params 中保留一次)

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

设置全局变量 state 用于保存当前状态，在 respond 获取 response 的同时更新状态，额外设置 policy_rules 字典表用于存放状态转换表，项目中设置了如下四种状态:

INIT, EMAIL, ADVICE, AFFIRMED = range(4) 分别表示初始状态，email 已认证/skip 状态，stock 限定调整状态，确定需要查询的 stock 的状态，给定的 policy_rules 即状态转换表如下：

```
policy_rules = {
    (INIT, "email"): (EMAIL, "Thank you for your subscription! Let's turn to get some stock Info now.", None),
    (INIT, "skip_email"): (EMAIL, "OK. Let's turn to get some stock Info now.", None),
    (INIT, "choose_stock"): (INIT, "Before your query, maybe you are concerned about more stock news.Perhaps you would be pleased to enter your email address to subscribe IEX stock news or simply click on 'skip_email'.", EMAIL),
    (INIT, "ask_stock_advice"):(INIT, "Before your query, maybe you are concerned about more stock news.Perhaps you would be pleased to enter your email address to subscribe IEX stock news or simply click on 'skip_email'.", EMAIL),
    (EMAIL, "choose_stock"):(ADVICE, None, None),
    (EMAIL, "ask_stock_advice"):(ADVICE, None, None),
```

```
        (ADVICE, "choose_stock"):(ADVICE, None, None),
        (ADVICE, "ask_stock_advice"):(ADVICE, None, None),
        (ADVICE, "affirm"):(AFFIRMED, "OK. Now you may choose the information you interested in." , None)
    }
```

当 response 为 None 时需要根据 params 和 neg_params 限定的条件查询数据库获取 stock。

状态转换处理方法，通过 spacy 中 interpreter 来获取 intent，依据(state,intent)对于不在状态转换表中的情况另行处理，若在表中则：

state, response, pending_state = policy_rules[(state, intent)]用于更新状态、获取 response、停等状态(见下说明)。

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

对于拒绝的处理，在 rasa-stock.json 中加入 deny 的 intent 并给定大量的实例从而识别否定的意图，项目中等待状态转换用于 user 在 skip/给定 email 之前询问 stock/请求 stock 建议的时候，设定等待状态为 ADVICE 表示已经完成第一次查询指定 stock/获取 stock 选取建议的选择，项目中对应代码处理为：

```
    state, response, pending_state = policy_rules[(state, intent)]
    if response is None:
    response, params, neg_params, suggestions, excluded = respond(message, params, neg_params, suggestions, excluded)
    if pending is not None and state == EMAIL:
    state, response1, pending_state = policy_rules[pending]
    if response1 == None:
        response1, params, neg_params, suggestions, excluded = respond(message, params, neg_params, suggestions, excluded)
    response = response + '\n' + response1
    if pending_state is not None:
        pending = (pending_state, intent)
        response0, params, neg_params, suggestions, excluded = respond(message, params, neg_params, suggestions, excluded)
```

此外仅在状态为 EMAIL 才会处理已经存在的 pending 行动，防止出现以外错误。

三、 项目总结

在本次项目中，老师为我们耐心详细讲解了如下知识点：

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. 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；

通过对知识点的融会贯通，我在问询股票信息为应用背景的金融聊天机器人的开发中熟悉了如何建立、开发、部署一个自己的 bot，锻炼了自身的代码能力、理解能力和文档查询、资料处理的能力，同时对于 NLP 领域、聊天智能机器人的研究背景、rasa_nlu 等用于自然语言处理的工具的发展进展有了初步的了解。

四、 参考资料

https://pypi.org/project/iexfinance/
https://core.telegram.org/api
https://spacy.io/
https://rasa.com/docs/rasa/nlu/using-nlu-only/
https://stackoverflow.com/questions/25338608/download-all-stock-symbol-list-of-a-market
https://dashboard.heroku.com/apps
deployment:
https://medium.com/@zaoldyeck9970/手把手教你怎麼打造-telegram-bot-a7b539c3402a
https://www.smartdeng.com/2019/06/24/flask_python-telegram-bot/