# 远程科研指导项目总结报告

姓名：王元铭

学校：北京工业大学

专业：信息安全

1 题目介绍

制作一个简单的对话机器人。功能是查询火车票信息。

2功能分析

总体来说要实现一个任务形对话机器人。主要可分为3步，

1. 理解用户的输入：即意图识别、关键信息提取
2. 与用户对话、对用户提问：根据前一个对话、用户的回复和现在已有的用户给出的信息三者生成下一个消息输出
3. 完成需要的任务

我需要做的是关于查询火车票的一个对话机器人。查询了其api后发现需要三个参数：出发地、目的地和出发时间。其中前两者是中文。也就是说我需要从用户的话中提取这三种信息。

在理解用户的输入这方面，因为只是一个简单的项目只有“查车票”一种意图。为了学习工具的使用强行加了“打招呼”和“道别”两种废话。这里使用rasa-nlu进行数据的训练（因为是意思意思训练数据也不多）。关键信息提取上，一是训练时标注了标签。二是在单独问出发地或目的地时可简单的用spacy的命名实体识别提取地点（“GPE”），若出现多个可分析他们的祖先中的介词来判断。

对于与用户对话。即引导用户回答api需要的三个关键数据。一开始告知用户本对话机器人的功能。给出示范。之后关键数据缺哪个问哪个，除了“打招呼”和“道别”和用户皮一下以外，毫不顾虑用户感受（毕竟只是一个简单的用来学习的项目）。三个关键数据有了之后把结果反馈给用户。

因为中文还要分词、现成的命名实体识别也不好找、训练数据貌似也更难搞（总而言之一是中文更复杂二是中文现成的东西少），所以本程序用的是英文进行对话。但火车票api要中文所以我还得翻译一下。翻译调用的网易的有道翻译的api（http://fanyi.youdao.com）。查地铁票用的急速数据的（https://www.jisuapi.com/）——括号内为网址

3具体实现

训练数据没啥好说的。直接调函数

trainer = Trainer(config.load("config\_spacy.yml"))

training\_data = load\_data('demo-rasa.json')

interpreter = trainer.train(training\_data)

在需要的存储数据上

state = {} # state存储出发地目的地和时间

message = '' # 存用户输入

respond = 'init' # 上一条机器人的输出的种类

last = '' # 上一条机器人的输出

下一个机器人的输出应当是关于state、message、respond的函数。但在用户输入废话或机器人听不懂的话时last用做最后的补救措施。

之后开始一轮一轮的询问、提信息、检查信息全了没、生成下一条询问…… 具体过程见源码。其中生成下一条询问的因为状态不多写进一个函数里了

其他的也没啥好说的。都在源码里写的很清楚。

4不足和改进

1. 对关键信息的识别不够准确。 改进方法：更多的训练数据、并且结合命名实体识别

对话优化。 改进方法：比如加入一些随机输出、一个问题多种问法。查询错误时（无效的地名或不正确的时间）语气委婉一点

1. 优化结果输出 改进方法：以表格形式输出结果，更加直观
2. 额外的功能 改进方法：额外添加排序、筛选功能（在表格的基础上）

5源码

# Import necessary modules

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

import sys

import json

import requests

import spacy

import numpy as np

import urllib3

import json

import urllib

import time

import urllib.request

# Create a trainer that uses this config

trainer = Trainer(config.load("config\_spacy.yml"))

# Load the training data

training\_data = load\_data('demo-rasa.json')

# Create an interpreter by training the model

interpreter = trainer.train(training\_data)

nlp = spacy.load("en\_core\_web\_md")

state = {} # state存储出发地目的地和时间

message = '' # 存用户输入

respond = 'init' # 存机器人说的话的种类

last = '' # 上一个回复

def my\_init():

global respond

global state

global last

global message

print("hello,I can help you search for train ticket information.")

respond = 'init'

last = "Please enter the place of departure and the time of arrival.(Can only inquire about nearly a month) \nFor example:\nI would like to check the train from Guangzhou to Xian on 2019-09-01."

print(last)

while 1==1:

input\_message() #开始对话

return None

def input\_message():

# Iterate over the word's ancestors

global message

message = input()

data = interpreter.parse(message)

if data['intent']['confidence'] >0.3:

message\_name = data['intent']['name']

else:

message\_name = "default"

#print(message\_name, data['intent']['confidence'])

respond\_message(message\_name)

# print(message\_name,data['intent']['confidence'])

# data['intent']['name']

return None

def respond\_message(message\_name):

global respond

global state

global last

global message

if message\_name == "goodbye":

state.clear()

respond = "goodbye"

last = "goodbye"

print(last)

time.sleep(3)

sys.exit()

# 对话结束

elif message\_name == "greet":

print("hello")

print("ummm......")

print(last)

#继续之前的对话

elif respond == "init":

if message\_name == "train\_info":

# print("111111")

get\_info() # 获取信息

check\_info() # 检测信息全不全

else:

print("......sorry,I can't understand")

print(last)

# 继续之前的对话

elif respond == "ask\_start":

doc = nlp(message)

gpe\_num = 0

for ent in doc.ents:

if ent.label\_ == 'GPE':

gpe\_num = gpe\_num + 1

if gpe\_num == 0:

print("......sorry,I can't understand")

print(last)

elif gpe\_num == 1:

for ent in doc.ents:

if ent.label\_ == 'GPE':

state['start\_place'] = ent.text

print("ok, the starting station is {}".format(state['start\_place']))

check\_info() # 检测信息全不全

else:

s\_word = []

che = 0

for ent in doc.ents:

if ent.label\_ == 'GPE':

s\_word.clear()

start\_list = list(doc[ent.start].ancestors)

for s\_l in start\_list:

s\_word.append(s\_l.text)

if 'from' in s\_word:

state['start\_place'] = ent.text

che = 1

if che == 1:

print("ok, the starting station is {}".format(state['start\_place']))

check\_info() # 检测信息全不全

else:

print("......sorry,I can't understand")

print(last)

elif respond == "ask\_end":

doc = nlp(message)

gpe\_num = 0

for ent in doc.ents:

if ent.label\_ == 'GPE':

gpe\_num = gpe\_num + 1

if gpe\_num == 0:

print("......sorry,I can't understand")

print(last)

elif gpe\_num == 1:

for ent in doc.ents:

if ent.label\_ == 'GPE':

state['end\_place'] = ent.text

print("ok, the ending station is {}".format(state['end\_place']))

check\_info() # 检测信息全不全

else:

s\_word = []

che = 0

for ent in doc.ents:

if ent.label\_ == 'GPE':

s\_word.clear()

end\_list = list(doc[ent.start].ancestors)

for s\_l in end\_list:

s\_word.append(s\_l.text)

if 'from' in s\_word:

state['end\_place'] = ent.text

che = 1

if che == 1:

print("ok, the ending station is {}".format(state['end\_place']))

check\_info() # 检测信息全不全

else:

print("......sorry,I can't understand")

print(last)

elif respond == "ask\_time":

if message\_name == "time\_info":

get\_info() # 获取信息

check\_info() # 检测信息全不全

else:

print("......sorry,I can't understand")

print(last)

# 继续之前的对话

elif respond == "end\_round":

respond = 'init'

last = "Please enter the place of departure and the time of arrival.(Can only inquire about nearly a month) \nFor example:\nI would like to check the train from Guangzhou to Xian on 2019-09-01."

print(last)

#继续之前的对话

else:

print("......sorry,I can't understand")

print(last)

return None

def get\_info():

global respond

global state

global last

global message

entities = interpreter.parse(message)["entities"]

#print(entities)

params = {}

# Fill the dictionary with entities

for ent in entities:

params[ent["entity"]] = str(ent["value"])

# print(params)

# print("haha")

for key, value in params.items():

state[key] = value

if 'month' in params.keys() and 'day' in params.keys():

print("setting {}-{} as date".format(state['month'], state['day']))

if 'start\_place' in params.keys():

print("setting {} as starting station".format(state['start\_place']))

if 'end\_place' in params.keys():

print("setting {} as ending station".format(state['end\_place']))

# 打印信息

return None

def check\_info():

global respond

global state

global last

global message

if 'start\_place' not in state.keys():

last = "Where is the starting place?"

respond = 'ask\_start'

print(last)

elif 'end\_place' not in state.keys():

last = "Where is the ending place?"

respond = 'ask\_end'

print(last)

elif 'month' not in state.keys():

last = "What is the date of departure? (example:2001-02-03)"

respond = 'ask\_time'

print(last)

elif 'day' not in state.keys():

last = "What is the date of departure?(example:2001-02-03)"

respond = 'ask\_time'

print(last)

else:

print("the starting station is {}\nthe ending station is {}\nthe time is {}-{}\n".format(state["start\_place"],state["end\_place"],state["month"],state["day"]))

print("Just a minute, please. Finding results")

jisuapi\_get\_reuslt()

return None

def youdao\_translate(word):

# 有道词典 api

url = 'http://fanyi.youdao.com/translate?smartresult=dict&smartresult=rule&smartresult=ugc&sessionFrom=null'

# 传输的参数，其中 i 为需要翻译的内容

key = {

'type': "AUTO",

'i': word,

"doctype": "json",

"version": "2.1",

"keyfrom": "fanyi.web",

"ue": "UTF-8",

"action": "FY\_BY\_CLICKBUTTON",

"typoResult": "true"

}

# key 这个字典为发送给有道词典服务器的内容

response = requests.post(url, data=key)

# 判断服务器是否相应成功

if response.status\_code == 200:

# 然后相应的结果

return response.text

else:

print("有道词典调用失败")

# 相应失败就返回空

return None

def youdao\_get\_reuslt(repsonse):

# 通过 json.loads 把返回的结果加载成 json 格式

result = json.loads(repsonse)

# print ("输入的词为：%s" % result['translateResult'][0][0]['src'])

# print ("翻译结果为：%s" % result['translateResult'][0][0]['tgt'])

return result['translateResult'][0][0]['tgt']

def jisuapi\_get\_reuslt():

global respond

global state

global last

global message

data = {}

data["appkey"] = "b6fa049c679c2e66" # 我自己的apikey

list\_trans = youdao\_translate(state['start\_place'])

data["start"] = youdao\_get\_reuslt(list\_trans)

list\_trans = youdao\_translate(state['end\_place'])

data["end"] = youdao\_get\_reuslt(list\_trans)

# t\_a = '{}-{}-{}'.format(time.strftime("%Y", time.localtime()), 1, 1)

# t\_b = time.strftime("%Y-%m-%d", time.localtime())

if time.localtime(time.time())[1]> int(state['month']) or( time.localtime(time.time())[1]== int(state['month']) and time.localtime(time.time())[2]> int(state['day'])):

data["date"] = '{}-{}-{}'.format(time.localtime(time.time())[0]+1, state['month'], state['day'])

else:

data["date"] = '{}-{}-{}'.format(time.localtime(time.time())[0], state['month'], state['day'])

url\_values = urllib.parse.urlencode(data)

url = "https://api.jisuapi.com/train/ticket" + "?" + url\_values

request = urllib.request.Request(url)

result = urllib.request.urlopen(request)

jsonarr = json.loads(result.read())

if jsonarr["status"] != 0:

print('search error:',jsonarr["msg"])

else:

result = jsonarr["result"]

print('{:15}\t{:15}\t{:15}\t{:15}\t{:15}'.format("trainno", "start station", "end station","departure time", "arrival time"))

for val in result["list"]:

if val["canbuy"] == 'Y':

print('{:15}\t{:15}\t{:15}\t{:15}\t{:15}'.format(val["trainno"], val["station"], val["endstation"], val["departuretime"], val["arrivaltime"]))

last ='Enter anything for the next round of queries'

respond = 'end\_round'

state.clear()

print(last)

my\_init()