telegram + rasa nlu + api实现智能聊天项目报告

0、 前言

1. 智能机器人接入平台选择

   开始选择的微信，但是我微信号无法接入wxpy接口

   转而现在telegram

2. telegram接入py包选择

   一开始选择的python-telegram-bot;但是该包调用url使用的urllib3，https支持极不友好；几经尝试，无法连接到telegram；

   转而选择pyTelegramBotAPI

3. 第三方api选择

   - rapidapi.com选择了covid-19的api接口，但是没调两下就被限制。（免费的调用次数及其有限）

```
def getCountries():
    url = "https://covid-19-data.p.rapidapi.com/help/countries"

    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122bxx280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)

    countries = [item['name'] for item in response.json()]
    return countries


def getLastTotals():
    url = "https://covid-19-data.p.rapidapi.com/totals"

    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122n3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    data = response.json()
```

```
    # s = 'the lastest situation of  COVID-19: deaths: {0};recovered: {1}'.format(data[0]['deaths'], data[0]['recovered'])
    return data

def getDailyReportTotals(date):
    url = "https://covid-19-data.p.rapidapi.com/report/totals"

    # querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": "2020-07-21"}
    querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": date}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)

    return json

def getDailyReportByCountryName(date, country):
    url = "https://covid-19-data.p.rapidapi.com/report/country/name"

    # querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": "2020-04-01", "name": "Italy"}
    querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": date, "name": country}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    return response.json()

def getLastestAllCountries():
    url = "https://covid-19-data.p.rapidapi.com/country/all"
    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    return response.json()

def getLastestCountryDataByName(name):
    url = "https://covid-19-data.p.rapidapi.com/country"

```

```
    # querystring = {"format": "json", "name": "italy"}
    querystring = {"format": "json", "name": name}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b7656-
sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers,
params=querystring)
```

故机器人在covid-19问答未完成调试

- 股票接口的选择
  一是iexfinance，二是twstock；

  iexfinance免费的情况下几乎没有接口可以调用（免费的也需要申请token）。故机器人在股票对话未调试
- 天气接口来自聚合数据

  机器人聊天完成天气查询、指定日期查询等；需要申请token；

1、摘要

本报告旨在记录聊天机器人项目过程情况，包括实现的一些问题，以及实现该项目的一些技术点、原理记录及梳理。

2、实现过程用到的技术点梳理

2.1 nlp

'聊天'就逃不开信息的传达，而文本信息的处理必然离不开自然语言处理（NLP），

NLP是数据科学的一大分支，旨在以智能而高效的方法来分析文本数据中潜藏的信息。采用适当的技术，你可以实现诸如自动总结，机器翻译，命名实体识别，情感分析等任务。

2.1.1 实体识别

对应实体的识别，单靠关键字匹配不具有一般性及通用性。可以使用spacy包加载数据，形成一个解析器。

2.1.2 nlp中的一些算法

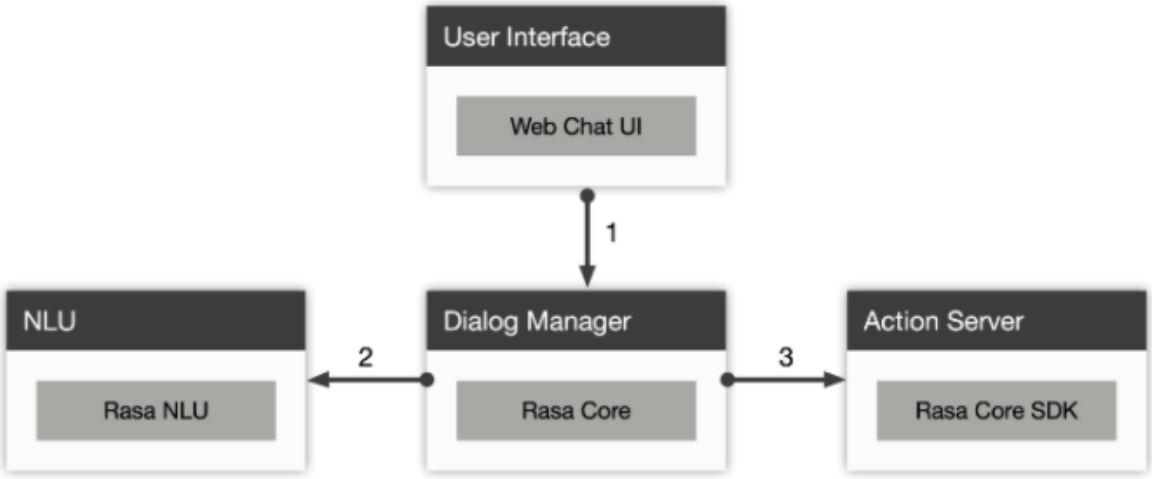
- 最近邻
- word2vec
- 支持向量机

2.1.3 状态机的使用

在多轮对话在，用来处理上下文。判断一轮对话是否结束，主要是多轮对话结合在一起抽取出所需信息。

2.2 rasa

open rasa source分rasa nlu和rasa core， 前者负责责理解用户的意图和提取相关的实体，后者负责管理整个对话的流程。有个比较完善的开源的天气查询机器人，实现技术Rasa 技术栈 （Rasa NLU, Rasa Core, Rasa Core SDK、python ui）



2.3 api的使用

- telegram
- stock
- weather

3、内容

本项目基于Rasa NLU实现意图识别部分（并为涉及TensorFlow、keras，这两者需要大量的数据集以及良好的计算机性能、最好是gpu，国内一般都是超算；才会训练处比较好的模型，从而给出良好的效果），并选择scikit-learn和SVM。

3.1 模型训练

```
from rasa_nlu.training_data import load_data
from rasa_nlu.model import Trainer
from rasa_nlu import config

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

# iexfinance
from iexfinance.stocks import Stock
from iexfinance.stocks import get_historical_data
from iexfinance.stocks import get_historical_intraday

import sqlite3,requests,time,re,random
from datetime import datetime
```

3.2 股票部分

```
# ------ 股票信息 ------
api_stock_token = 'pk_276fff8a2846b3b0704bf4e86b24db'
```

```
3
4
5  # from twstock import Stock
6  # import twstock
7  # 获取某股票的当前价格
8  def get_current_price(company):
9      print("Company: ", company)
10
11     prices = Stock(company, token=api_stock_token).get_price()
12     # prices = Stock(company)
13     return prices
14
15
16 # 获取某股票的每股利润
17 def get_ttmEPS(company):
18     ttmEPS = Stock(company).get_key_stats()['ttmEPS']
19     return ttmEPS
20
21
22 # 获取某股票相关新闻
23 def get_news(company):
24     # news = twstock.realtime.get(company)
25     news = Stock(company, token=api_stock_token).get_news()
26     if news:
27         return str(news)
28     for i in news:
29         if i['summary'] != 'No summary available.':
30             return i['url']
```

3.2.1 股票数据绘图

```
1  # 生成历史数据折线图
2  def generate_figure(message):
3      comprehended_data = interpreter.parse(message)
4
5      for i in range(0, 2):
6          # 获取公司名称
7          if comprehended_data['entities'][i]['entity'] == 'company':
8              required_company = comprehended_data['entities'][i]['value']
9
10         # 获取数据类型 open / close / high
11         if comprehended_data['entities'][i]['entity'] == 'his_price_type':
12             required_type = comprehended_data['entities'][i]['value']
13
14         # 默认值
15         else:
16             required_company = 'AAPL'
17             required_type = 'close'
18
19     # 对模糊的历史数据询问的补充信息
20     if len(comprehended_data['entities']) <= 3:
21
22         # 时间格式：2019-1-1
23         time_period = [comprehended_data['entities'][0]['value'],
24                        comprehended_data['entities'][1]['value']]
25
26         start_time_splited = time_period[0].split(' - ')
```

```
        end_time_splited = time_period[1].split(' - ')

        # 开始时间
        # print(start_time_splited)

        start_year = int(start_time_splited[0])
        start_month = int(start_time_splited[1])
        start_day = int(start_time_splited[2])

        # print("Start year: ", start_year, ", Start month: ", start_month, ", Start day", start_day)

        # 结束时间
        end_year = int(end_time_splited[0])
        end_month = int(end_time_splited[1])
        end_day = int(end_time_splited[2])

        start_time = datetime(start_year, start_month, start_day)
        end_time = datetime(end_year, end_month, end_day)

        # 生成该时间段的线形图
        his_data = get_historical_data(required_company, start_time, end_time, output_format='pandas')
```

3.3 天气部分

```
week = {'Monday': 1, 'Tuesday': 2, 'Wednesday': 3, 'Thursday': 4, 'Friday': 5, 'Saturday': 6, 'Sunday': 0}


def get_weekday(message):
    # 匹配询问的星期
    weekday = re.findall("[A-Z]+[a-z]*", message)

    # 没有星期，默认为今天
    if weekday == []:
        return [0]

    else:
        # 今天的星期
        today = int(time.strftime("%w"))

        # api中要查找的列数
        number = []

        for day in weekday:
            n = week[day] - today
            if (n < 0):
                n = n + 7
            number.append(n)

        return number


```

```python
# ------ 在数据库中查省份代号（用于天气api） ------

def get_citycode(city):
    conn = sqlite3.connect('city_code.db')
    c = conn.cursor()

    code = ''

    query = "SELECT * FROM city WHERE name = '" + city + "'"
    c.execute(query)
    result = c.fetchall()

    for row in result:
        code = row[0]

    return code


# ------ 调用api返回各省天气信息 ------
def get_weather(day_list, city):
    # 申请一个key：https://www.juhe.cn/docs/api/id/39
    weather_key = "41b7b79a48958b766f756debe3860077"

    # 省份编号
    code = get_citycode(city)

    url = "http://v.juhe.cn/weather/index?format=2&cityname=" + code + "&key=" + weather_key
    req = requests.get(url)
    info = dict(req.json())
    info = info['result']['future']
    # print(info)

    response = ""

    for number in day_list:
        newinfo = info[number]
        temperature = newinfo['temperature']
        weather = newinfo['weather']
        wind = newinfo['wind']
        week = newinfo['week']
        date = newinfo['date']
        response = response + "日期: " + date + " " + week + ", 温度: " + temperature + ", 天气: " + weather + ", 风向与风力: " + wind + "\n"

    return response


def get_deny_weather(day_list, city, message):
    # print("old: ", day_list)

    # 匹配询问的星期
    weekday = re.findall("[A-Z]+[a-z]*", message)

    # 今天的星期
    today = int(time.strftime("%w"))

    # 移除否定的星期
```

```
    for day in weekday:
        n = week[day] - today
        if (n < 0):
            n = n + 7
        day_list.remove(n)

    # print("new: ", day_list)

    return get_weather(day_list, city)
```

3.4 COVID-9

```
def getCountries():
    url = "https://covid-19-data.p.rapidapi.com/help/countries"

    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122bxx280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)

    countries = [item['name'] for item in response.json()]
    return countries


def getLastTotals():
    url = "https://covid-19-data.p.rapidapi.com/totals"

    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122n3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    data = response.json()
    # s = 'the lastest situation of  COVID-19: deaths: {0};recovered: {1}'.format(data[0]['deaths'], data[0]['recovered'])
    return data

def getDailyReportTotals(date):
    url = "https://covid-19-data.p.rapidapi.com/report/totals"

    # querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": "2020-07-21"}
    querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": date}

```

```
    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)

    return json

def getDailyReportByCountryName(date, country):
    url = "https://covid-19-data.p.rapidapi.com/report/country/name"

    # querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": "2020-04-01", "name": "Italy"}
    querystring = {"date-format": "YYYY-MM-DD", "format": "json", "date": date, "name": country}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    return response.json()

def getLastestAllCountries():
    url = "https://covid-19-data.p.rapidapi.com/country/all"
    querystring = {"format": "json"}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b765-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
    return response.json()

def getLastestCountryDataByName(name):
    url = "https://covid-19-data.p.rapidapi.com/country"

    # querystring = {"format": "json", "name": "italy"}
    querystring = {"format": "json", "name": name}

    headers = {
        'x-rapidapi-host': "covid-19-data.p.rapidapi.com",
        'x-rapidapi-key': "d4f5fcf3d9msh84a9122xx3280bdp1b7656-sn39390fd68eb1"
    }

    response = requests.request("GET", url, headers=headers, params=querystring)
```

3.4 接入telegram

```
import telebot

API_TOKEN = '1173381475:AAEdk-i625dOvhkbFvOjyv3-iNpJOSCMLWI'
bot = telebot.TeleBot(API_TOKEN)


# # Handle '/start' and '/help'
@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    bot.reply_to(message, """\
what I can do Currently : \n1. Get stock information \n    \
1.1 Get current data \n    1.2 Get historical data \n    1.3 Analyze certain stocks \n\
2. Get weather information(every provience in China, seven days)\n \
3. Get the info of COVID-19(hasn't opened)
and you can send '/help' to get those infomation"
I am here to echo your kind words back to you. Just say anything nice and I'll say the exact same thing to you!\
""")


# Handle all other messages with content_type 'text' (content_types defaults to ['text'])
@bot.message_handler(func=lambda message: True)
def echo_message(message):
    state = MAIN
    pending = None
    data = {'message': {'text': str(message.text), 'id': str(message.chat.id)}}
    state, pending, final_response, message_intent = send_message(state, pending, str(message.text))
    if message_intent == 'clear_historical_data' or message_intent == 'add_historical_data':
        try:
            photo = open('img.png', 'rb')
            bot.send_photo(str(message.chat.id), photo)
            bot.send_photo(str(message.chat.id), "FILEID")
            photo.close()
        except:
            bot.reply_to(message, 'sorry,the function of sending img broken!')
    else:
        bot.reply_to(message, final_response)


bot.polling()
```

4、总结

该报告总结梳理了聊天机器人项目的技术点和实现过程；并介绍了聊天机器人的实现提供库存和天气信息。这是一个非常简单的聊天机器人，但对于新手来说是一个很好的练手机会，即巩固复习了学到的知识点，又通过项目的实施查找了许多相关资料。一直以来，人工智能都有极大的发展前景，智能聊天机器人更是隐藏了巨大潜力，我们可以希望它能进一步提高人们的工作效率。

但这也显示了提高聊天机器人准确性的困难。一些聊天机器人创建Siri和Cortana等高科技公司在实际应用中表现出色，但是它们还是无法像人一样。苹果和微软等公司持有大量股份数据集和高性能设备，但其聊天机器人的性能还是不够好，相关算法仍然有待改进。庞大的训练数据集很重要。当我们忙于完善数据集并改善现有算法的参数，我们

还应该记住探索新方法。我从这堂课中学到了很多东西。

在这节课之前，我只知道深度学习方法用于处理自然语言。我曾经尝试使用LSTM处理数据，性能不是很好。我在这堂课中了解了sklearn和SVM，它们是擅长处理小规模数据。我希望我能学到更多关于人造的知识情报和自然语言处理领域。

感谢我的导师的帮助和宽容。