金融类智能聊天机器人

远程科研项目

课程名称：金融类智能聊天机器人

专　　业：计算机科学与技术

学生姓名：李彦澔

2019 年　　8 月　　14 日

目录

摘要

随着互联网与人工智能的发展，人们切实的体会到了高新技术带来的便利，各种智能的产品使得计算机能够更精准，更方便的获取用户的需求，用户也从而节省了许多机械化工作的时间。本项目的设计背景与目的则与此一致，即让用户以尽可能自然，贴近人类交流行为的方式来获取他们所需要的信息。

本项目的最主要的应用场景为金融背景，在获取股票所需要的信息时，通常需要通过复杂的专业软件来进行查询，且用户时常要与图表打交道，因此，刚刚入门的新手往往不清楚如何来操作这些软件，更不用说看股票，记股票，选股票了。因此，针对这些希望投身金融业，注重应用且希望快速入门的用户，在此类人群的工作场景中，本项目能够较好的满足此部分人群的需求。

对于本项目，使用的编程语言为 python，使用的最主要的工具为 rasa_nlu，spacy，telegram。实现的主要核心技术为：基于正则表达式的格式化回答；通过关键词，正则表达式，spacy，和 rasa_nlu 等多种技术融合的意图识别和实体抽取；甄别否定实体技术；基于 sqlite3 的本地数据库查询技术；基于状态机和等待状态转换的含遗留任务且可处理拒绝状态的多轮多次查询技术；基于 iexfinance api 的股票信息查询；基于百度 api 的语音识别技术。在本地测试完毕后使用 telegram 的开源代码将其集成到了 telegram bot 上使其成为一个真正的聊天机器人。

从整体的逻辑上看，本项目分为登录前和登录后两个部分，在登录前，可以和机器人进行一些闲聊等不涉及股票数据上的交流。在登录完成后，用户可以通过多轮查询来查询自己所需要的信息，由于查询的信息都是基于特定股票的，因此对于大部分的查询都需要先选定自己想查询的股票，然后才能进行后续操作。

登录后，面对一些对股票不熟悉的用户，可以首先让机器人给自己推荐一些股票，通过查询本地数据库来返回结果。使用本地数据库的原因是方便量化每个股票的特征，同时也给推荐算法的开发预留了空间。

在得知了自己想查询的股票后，用户可以通过多轮查询来一步一步让机器人获取用户想查询的信息，对于其中的历史价格查询，由于数据量较大，因此可以选用以表格方式来回复用户，以文件方式发送。

为了满足多种使用场景，例如日常的生活场景以及专业的工作场景，本机器人提供了两种交互方式，分别是语音交互以及文字交互，二者无显式的模式切换，用户可自由使用文本框或者语音信息按钮来发送信息，机器人能够自由识别对应的格式并加以处理。

关键词：SPACY RASA_NLU TELEGRAM 数据库查询 语音查询 状态机 否定甄别 待定行动 多轮多次查询

一、背景

1.1 聊天机器人

聊天机器人（Chatterbot）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（如 Google 智能助理）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（Facebook Messenger）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。

聊天机器人（chatterbot）是一个用来模拟人类对话或聊天的程序。“Eliza”和 “Parry”是早期非常著名的聊天机器人。它试图建立这样的程序：至少暂时性地让一个真正的人类认为他们正在和另一个人聊天。

聊天机器人产生的原因是，研发者把自己感兴趣的回答放到数据库中，当一个问题被抛给聊天机器人时，它通过算法，从数据库中找到最贴切的答案，回复给它的聊伴。

此外，聊天机器人的成功之处在于，研发者将大量网络流行的俏皮语言加入词库，当你发送的词组和句子被词库识别后，程序将通过算法把预先设定好的回答回复给你。而词库的丰富程度、回复的速度，是一个聊天机器人能不能得到大众喜欢的重要因素。千篇一律的回答不能得到大众青睐，中规中矩的话语也不会引起人们共鸣。此外，只要程序启动，聊士们 24 小时在线随叫随到，堪称贴心之至。

1.2 股票查询

随着国内证券分析技术和软件技术的不断提升，如今的股票软件更加的实用化、功能化、智能化，从动态行情分析，实时新闻资讯，智能选股，委托交易等方面做了更深的研究，使得广大的股民朋友在基本分析、技术分析，新闻资讯汇集、个性选股、自动选股、自动委托交易，止赢止损等等方面获得更快更全更好的服务，争取最大程度的赢利。

国内证券软件开发商也根据客户的不同要求开发出了种种不同特点的股票软件产品，比较有特色的的版本有同花顺，大智慧，通达信。

股票软件的实质是通过对市场信息数据的统计，按照一定的分析模型来给出

数（报表）、形（指标图形）、文（资讯链接），用户则依照一定的分析理论，来对这些结论进行解释，也有一些傻瓜式的易用软件会直接给出买卖的建议。其实，比较正确，或者实在的用法，是应该挑选一款性能稳定、信息精准的软件，结合自己的炒股经验，经过摸索之后，形成一套行之有效的应用法则，那样才是值得信赖的办法，而机械地轻信软件自动发出的进场离场的信号，往往会谬以千里。

二、任务分析以及思路整理

本项目的目的即构建一个可进行金融查询的聊天机器人，实现的目标应当是高鲁棒性。

经老师对各个知识点的讲解与梳理，初期可以将该工程分为以下几步：

1. 对于输入的信息进行意图识别，判断来访者的真实用意。

2. 根据来访者的具体意图，来进行意图中的命名实体。

3. 得到了二者后，应当使用实体来进行数据库或者接口的查询和访问获得所需信息。

4. 根据信息类型，大小，特点的不同，进行不同的预处理后，以合适的形式反馈给用户。

然而，在初期实践过程中，我发现，仅仅使用以上四步虽然涵盖的范围较为广泛，但是效果并不尽人意，主要原因是，对于意图为查询金融信息的消息，往往含有一些固定的关键字，且一般消息的格式也比较固定，但是对于闲聊类的消息，往往变化多端，且形式极其灵活。在这种情况下，死板的带入以上四步往往会使得机器人误判意图或者实体抽取错误。

显然，在工业环境下，仅仅迷信训练好的模型无法使得使用者拥有优质的用户体验，因此为了使得机器人能够更加的灵通，我在这四步的基础上加入了机器人中较为“死板”以及“套路”的聊天选项。对于一些拥有固定格式或者实体意图不明确的消息，使用闲聊情景下的万金油型回复来“搪塞”用户，在这种情形下，这样虽然不能立即回答用户的问题，但是能够保持机器人对话流畅性的情况下提醒用户调整自己问询语句的格式。

因此，经以上分析，整体逻辑应为：在消息送至机器人后，应当首先进行意图抽取，跟据意图的抽取情况来判断下一步的行为，由于对于一些闲聊性的消息，经过训练模型的预测，仍旧可能存在其以较高的置信度被识别为查询类消息的情况，此时，因此闲聊类消息的处理优先级应当大于查询类消息，即意图抽取后，应当首先导入聊天类消息的函数，若在函数中判断其并不符合聊天类消息的形

式，再导入查询类消息的函数。

若消息成功进入了查询类消息的处理部分，则一切都变得明朗了，对于该类信息，由于其消息存在前后相关的严密逻辑性，因此，对于该项目的应用场景，使用状态机来控制此类消息的交互是再合适不过的了，同时，由于存在着类似登录或者前置信息的情况，因此该状态机应当有等待任务转换，遗留任务处理以及拒绝肯定回复的处理机制。

以上即为本项目在算法层面上的大致思路，为了提高其应用性，我尝试将其移植到社交软件上，经过大量尝试，最终选择了 telegram。原因如下：

1. telegram 的开源性极高，不光可以实现一个 bot，用户甚至可以根据开源文档设计一个属于自己的专属 telegram。

2. 微信对于此类行为管控较严格，使用时经常会导致网页版微信被封禁的情况。

3. telegram 的相关开发资料较为丰富，生态圈较为成熟，虽然其绝大多数为外文文献，但是仔细阅读后能够迅速掌握该软件各种构造类以及其具体使用方式。

4. telegram 处理各类消息的方式较为简便，对于一些比较长或者格式化的消息，可以使其以表格，图片或者文件等形式回复。

除此之外，为了提高查询的灵活性，本项目加入了语音查询的功能，用户不需要显式的声明自己查询的模式，若需要语音查询，则直接通过 telegram 的语音信息发送按钮发送自己想查询的信息即可。

三、项目工程的详细设计与代码实现

3.1 意图识别和实体抽取

根据以上步骤的设计与分析，以下的任务即为选择适合的工具与好的方法来进行代码实现，对于本项目，我使用的最主要的工具是 spacy 和 rasa_nlu，同时以正则表达式加以辅助。

对于意图识别，本项目采用了模型预测，正则表达式和关键字等结合的方法来进行，其中模型预测使用的为 rasa_nlu 和 spacy。通过训练自己配置的语料库来得到对应的模型，然后使用 interpreter.load()加载，即可操作该类中的各种功能，包括但不限于实体抽取和意图识别。

```
#训练数据
interpreter=Interpreter.load('D:\Chat_robot\default\model_20190811-205136')
print('ok')
```

图 1 加载训练好的模型

图 2 训练集示例

如任务分析阶段所述，只依靠于训练模型得到结果的精确度是远远不够的，对于一些闲聊类的，实体和意图不明显的消息，应当设置特殊的处理方式来保证机器人的对话流畅性。

由此，本项目中采用了关键词，正则表达式的来辅助机器人进行回复，实现方式是使用字典来存储一系列意图以及意图对应的关键词。

```
{
'greet': [' hello ', ' hi ', ' hey '],
'thankyou': ['thank', 'thx'],
'goodbye': ['bye', 'farewell','that is all','ok,thanks'],
```

图 3 关键词字典示例

除以上情景外，还存在有格式化消息的情况，对于这种消息，我采用了固定套路回复的方式来处理，对于这种情况，回复的消息往往都是没什么营养的“万金油”式，不过作为机器人的最后防线，这种方式往往效果还不错，能够有效保证机器人对话的流畅性，同时也能提醒用户更换自己问问题的形式。

```
rules = {'I wanna (.*)': ['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}",
                                 ],
         'do you think (.*)': ['if {0}? Absolutely.'],
         'if (.*)': ["Do you really think it's likely that {0}",
                     'Do you wish that {0}',
                     'What do you think about {0}',
                     'Really--if {0}'],
         'Im (.*)': ["hello {0}"
                     ],
         'my name is (.*)': ["hello {0}"
                ]
         }
```

图 4 固定套路回复示例

若机器人的最后一道防线被突破，则只能抛出“我听不懂你在说什么”的消息了。

3.2 查询类消息的处理

进入到该部分的唯一途径就是在意图识别和实体抽取环节，用户的信息被识别为查询类消息，且实体明确，否则机器人永远不会执行该环节所提及的操作。

由于对于不同特点的消息，要采取不同的策略去处理，多变的聊天类消息应当用较为自由的方式去处理，但是一旦进入到了功能查询部分，由于该部分的信息存在有连贯性和逻辑性的要求，则需要此部分的操作应当较为刻板。

对于这种刻板的，格式化的操作，本项目采用了状态机来处理，在此基础上，本项目实现了带遗留任务，状态等待转换，拒绝甄别与处理的多轮多次查询。

```
  (INIT, "stock_search"): (INIT, "please enter your password?", AUTHED),
  (INIT, "location"): (INIT, "please enter your password first.", AUTHED),
```

图 4 状态机示例

首先在该类信息的条件下，为了控制请求接口的 QPS 以及安全性，对于任何的查询类消息，都需要经过登录验证操作，在该情况下，只有当验证通过后，才能够进行下面的查询操作，否则，将会持续的轮循该步骤。

在登录验证完成后，通过更改程序的登录标志位，来使得该用户进入已登录的状态，用户可以根据自己的需求来查询自己所需要的信息，查询的具体实现方式为请求接口。

```
def get_book(ID):
    appl = Stock(ID, token = TK)
    x=appl.get_book()
    return x
```

```
def get_historical_prices(ID):
    appl = Stock(ID, token = TK)
    x=appl.get_historical_prices()
    return x
```

```
def get_previous_day_prices(ID):
    appl = Stock(ID, token = TK)
    x=appl.get_previous_day_prices()
    return x
```

```
def get_quote(ID):
    appl = Stock(ID, token = TK)
    x=appl.get_quote()
    return x
```

图 5 请求接口调用方式示例

3.3 股票推荐请求的处理

有时，用户的需求可能并不是查询一些特定股票的相关信息，他们可能仅仅是想获得一些系统推荐的股票，来作为自己的投资参考，在这种情况下，通过请求接口的方式去获得股票名字并不合适，原因如下：

1. 请求接口的速度一般比较慢，对于该种返回信息比较少的消息类型并不是最优解。

2. 由于用户查询时往往会夹杂有一些个人的查询意愿以及对所查询股票的特征描述，这时，使用接口来查询便显得低效率。

综上，结合该类消息的相关特点，使用数据库的方式在该种情况下效果比调用接口要好。

由于在进行该部分操作前，已经经历过实体抽取和意图识别，因此，此时，用户发送的消息中已经包含了所有数据库查询的必要条件，因此，此时只需要将相应的实体匹配上意图来生成 SQL 查询语句便可实现从本地数据库中进行信息查询。

```
query = 'SELECT * FROM stocklist'
if len(params) > 0:
    filters = ["{}=?".format(k) for k in params] + ["name!='?'".format(k) for k in excluded]
    query += " WHERE " + " and ".join(filters)
t = tuple(params.values())

# open connection to DB
conn = sqlite3.connect('stocklist.db')
# create a cursor
c = conn.cursor()
c.execute(query, t)
```

图 5 由意图和实体生成查询语句的部分代码示例

相比于接口调用的方式，本项目选用数据库的原因如下：

1. 数据库的查询速度比接口的请求速度要快很多，用户体验上使用数据库要远优于使用接口。

2. 数据库中可以自由添加自定义的各种特征以及要求，由此可以更好的满足用户的个性化查询。

3. 可以为查询的推荐算法提供较大的开发空间，使得用户能够优先查询到用户想看到的，亦或是公司想让用户看到的一系列股票。

3.4 语音请求

为了更加方便用户的操作，此项目中包含了语音查询的功能，用户不需要显式的声明查询模式，若需要语音查询，则直接通过 telegram 的发送语音信息按钮即可查询，此部分使用的是百度 api 的 sdk 方式。

```
def get_voice_text():
    # 识别本地文件
    x=client.asr(get_file_content('result.pcm'), 'pcm', 16000, {'dev_pid': 1737,})
    print(x)
    x=x["result"][0]
    return x
```

图 6 调用语音接口部分代码示例

3.5 回复消息的预处理

对于一些数据量比较庞大的信息，例如历史价格等等，在这种情况下，通过简单的文本回答往往会使得聊天框被占满，不美观，此外，也有可能因为消息过长而使得回复根本发不出去，在这种情况下，需要选用适合的形式来回复消息，例如图片，表格等等。

因此，对于一些比较庞大的信息，需要加以判断，然后处理返回的 json 串，按照格式生成相关文件来返回。

此部分调用的是 python 自带的对 excel 表格处理的 openpyxl 库。

3.6 telegram 的整合

对于该项目，我用到的 telegram 中的库和构造类有 Updater，CommandHandler，MessageHandler，Filters，InputFile。

解释与功能如下：

Updater：用于实现机器人与使用用户的交互，通过该构造类能够认证机器人及其所有者。

CommandHandler：用于向机器人发送控制类的命令，例如/start 为激活机器人。

MessageHandler：为消息的交互类，是本项目中该部分最重要的类。

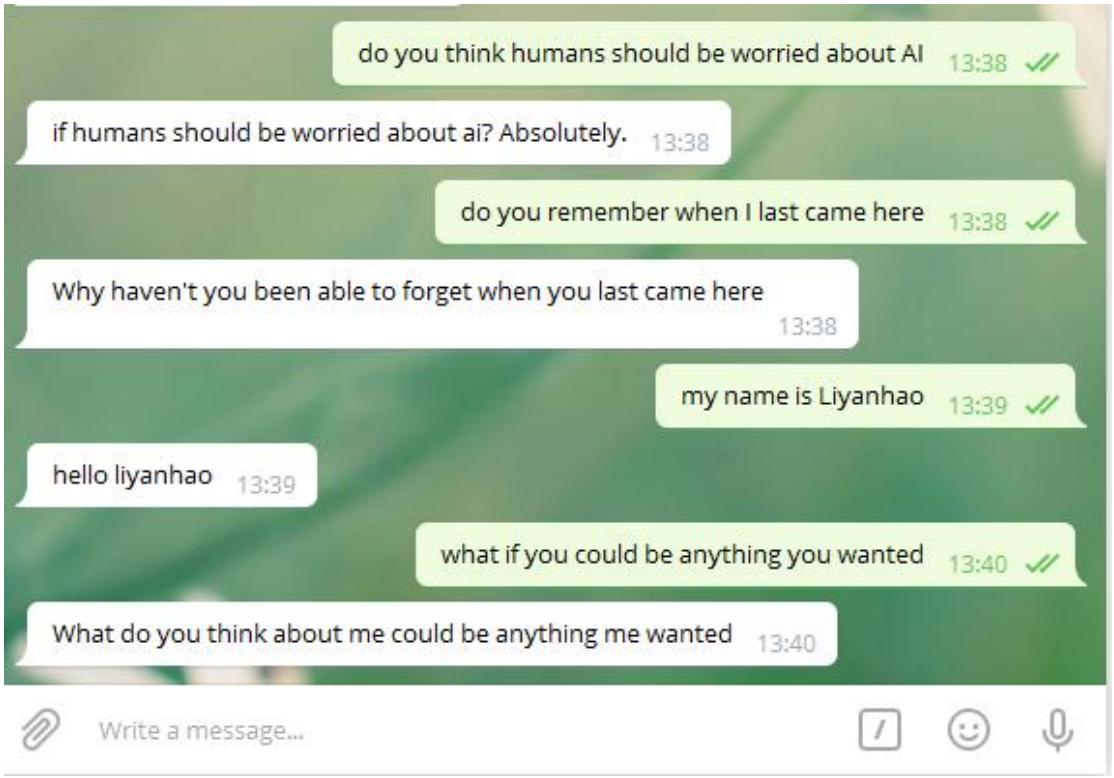
Filters：用于实现消息类型的检测，例如 voice 类，image 类，document 类等等。

InputFile：用于机器人向用户发送文件。

四、成果展示

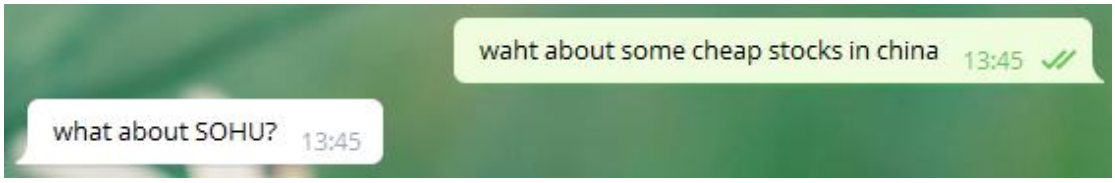
4.1 闲聊信息的展示

4.2 格式化回复信息的展示



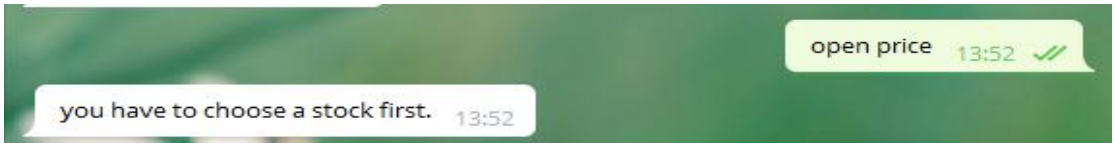
4. 3 在未登录情况下请求股票推荐

4. 4 在已登录情况下请求股票推荐

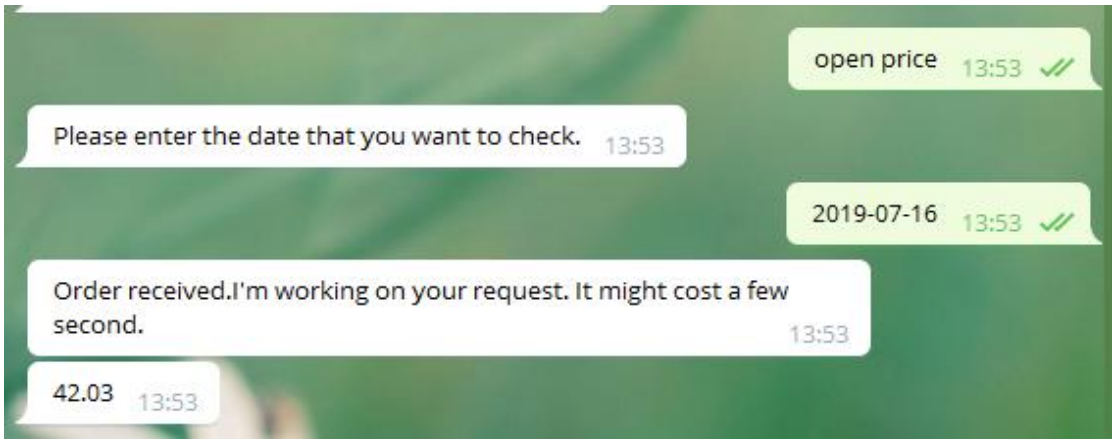


4. 5 确认需要查询的股票

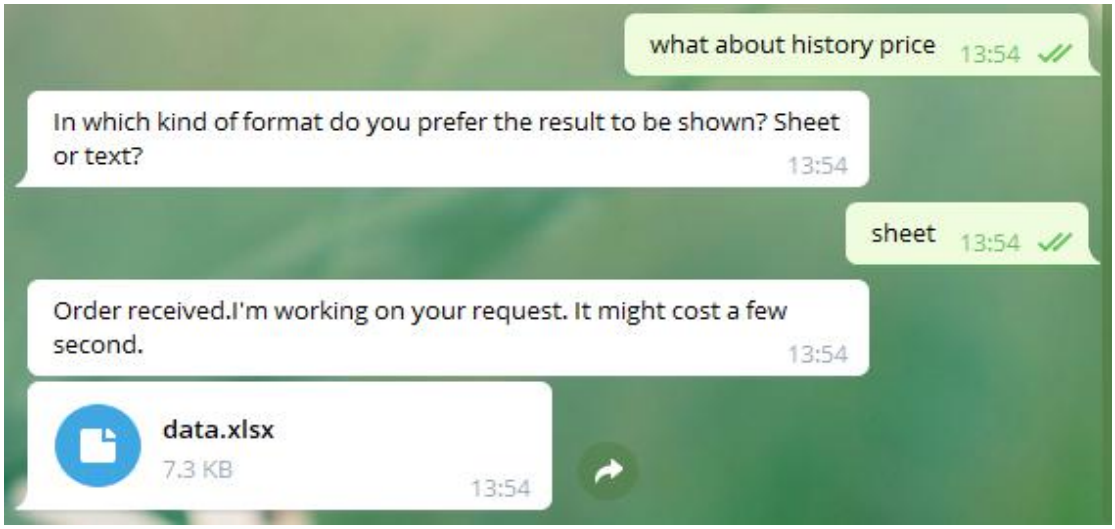
4. 6 在未选择股票的情况下查询特定信息



4. 7 查询信息举例

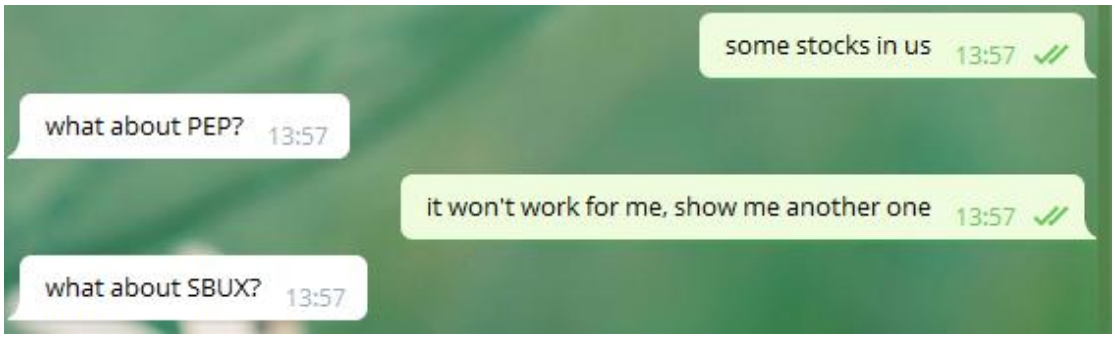


4. 8 查询表格举例

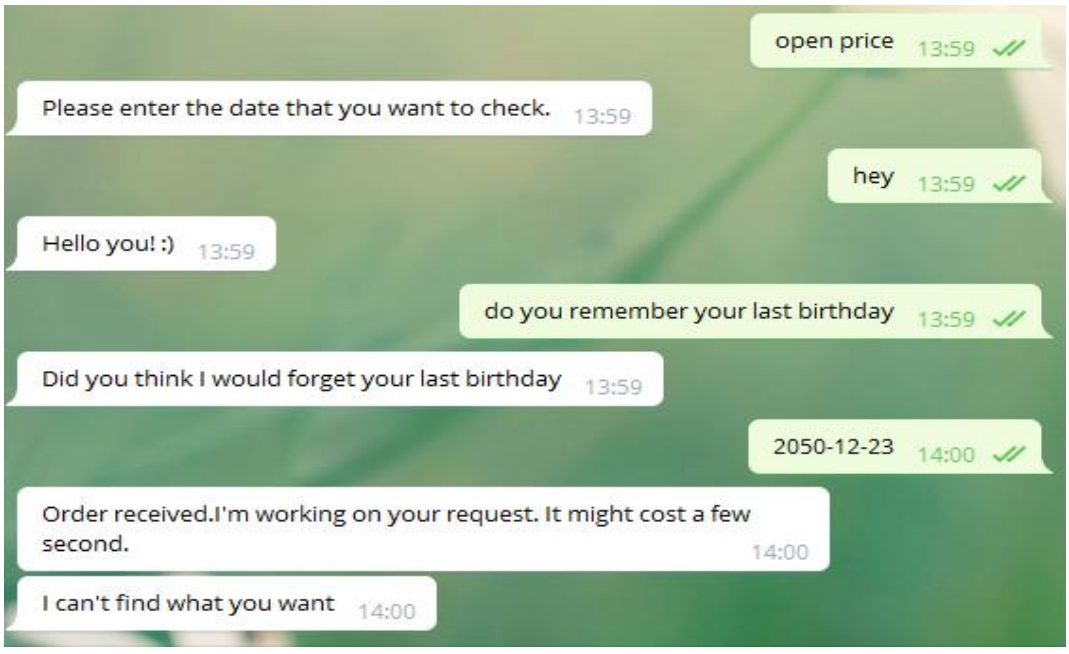


change	ngeOverT	ngePerc	close	date	high	label	low	open	uClose	uHigh	uLow	uOpen	uVolume	volume
0	0	0	42.29	2019-07-1	42.6	Jul 15	41.33	41.61	42.29	42.6	41.33	41.61	872170	872170
-0.21	-0.00497	-0.4966	42.08	2019-07-1	42.75	Jul 16	41.77	42.03	42.08	42.75	41.77	42.03	519835	519835
-1.13	-0.03169	-2.6854	40.95	2019-07-1	42.14	Jul 17	40.92	42.08	40.95	42.14	40.92	42.08	260624	260624
-0.88	-0.0525	-2.149	40.07	2019-07-1	41.01	Jul 18	39.88	41.01	40.07	41.01	39.88	41.01	613783	613783
0.37	-0.04375	0.9234	40.44	2019-07-1	41.02	Jul 19	40.42	40.53	40.44	41.02	40.42	40.53	662329	662329
-0.56	-0.05699	-1.3848	39.88	2019-07-2	40.69	Jul 22	39.56	40.44	39.88	40.69	39.56	40.44	902936	902936
0.73	-0.03973	1.8305	40.61	2019-07-2	41.3	Jul 23	40.19	40.19	40.61	41.3	40.19	40.19	1013273	1013273
0.41	-0.03003	1.0096	41.02	2019-07-2	41.04	Jul 24	40.42	40.75	41.02	41.04	40.42	40.75	274060	274060
-0.65	-0.0454	-1.5846	40.37	2019-07-2	41.33	Jul 25	40.04	41.33	40.37	41.33	40.04	41.33	454776	454776
-0.04	-0.04635	-0.0991	40.33	2019-07-2	40.75	Jul 26	40.03	40.65	40.33	40.75	40.03	40.65	460592	460592
0.03	-0.04564	0.0744	40.36	2019-07-2	40.73	Jul 29	40.06	40.29	40.36	40.73	40.06	40.29	619979	619979
-0.28	-0.05226	-0.6938	40.08	2019-07-3	40.35	Jul 30	39.47	39.9	40.08	40.35	39.47	39.9	492151	492151
-0.96	-0.07496	-2.3952	39.12	2019-07-3	40.05	Jul 31	38.63	40.05	39.12	40.05	38.63	40.05	865797	865797
-1.76	-0.11658	-4.499	37.36	2019-08-0	39.51	Aug 1	37.11	39.15	37.36	39.51	37.11	39.15	1064391	1064391
-0.06	-0.118	-0.1606	37.3	2019-08-0	37.77	Aug 2	36.57	37.07	37.3	37.77	36.57	37.07	629357	629357
-2.89	-0.18633	-7.748	34.41	2019-08-0	35.76	Aug 5	33.95	35.76	34.41	35.76	33.95	35.76	1208231	1208231
0.82	-0.16694	2.383	35.23	2019-08-0	36.04	Aug 6	35	35.09	35.23	36.04	35	35.09	1201426	1201426
-0.02	-0.16742	-0.0568	35.21	2019-08-0	35.6	Aug 7	34.4	34.66	35.21	35.6	34.4	34.66	1013158	1013158
0.5	-0.15559	1.4201	35.71	2019-08-0	36	Aug 8	35.28	35.59	35.71	36	35.28	35.59	508580	508580
-1.19	-0.18373	-3.3324	34.52	2019-08-0	35.4	Aug 9	34.52	35.34	34.52	35.4	34.52	35.34	1149580	1149580
0.91	-0.16221	2.6362	35.43	2019-08-1	35.5	Aug 12	33.95	34.01	35.43	35.5	33.95	34.01	817863	817863
0.36	-0.1537	1.0161	35.79	2019-08-1	36.54	Aug 13	32.99	34.66	35.79	36.54	32.99	34.66	1076966	1076966
-0.64	-0.16883	-1.7882	35.15	2019-08-1	35.49	Aug 14	34.58	34.99	35.15	35.49	34.58	34.99	738363	738363
0.51	0	1.43	35.66	2019-08-1	36.28	Aug 15	35.31	35.92	35.66	36.28	35.31	35.92	1290437	1290437

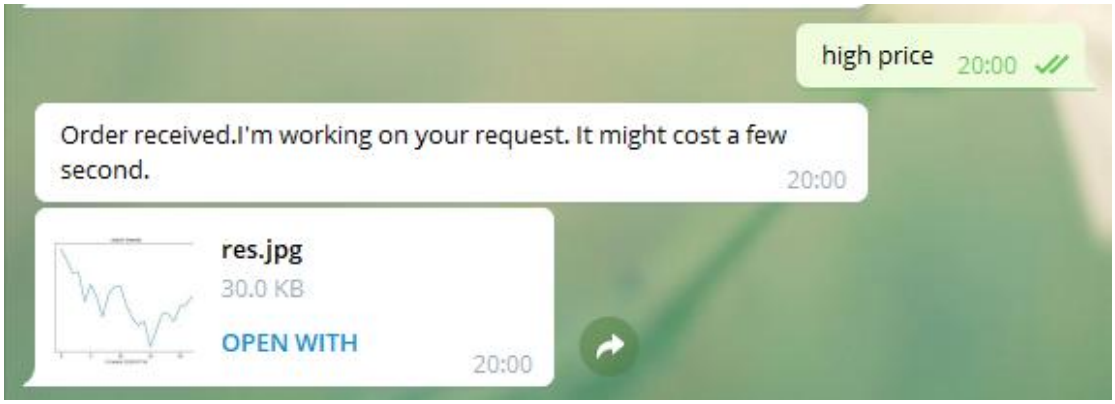
4.9 否定甄别举例

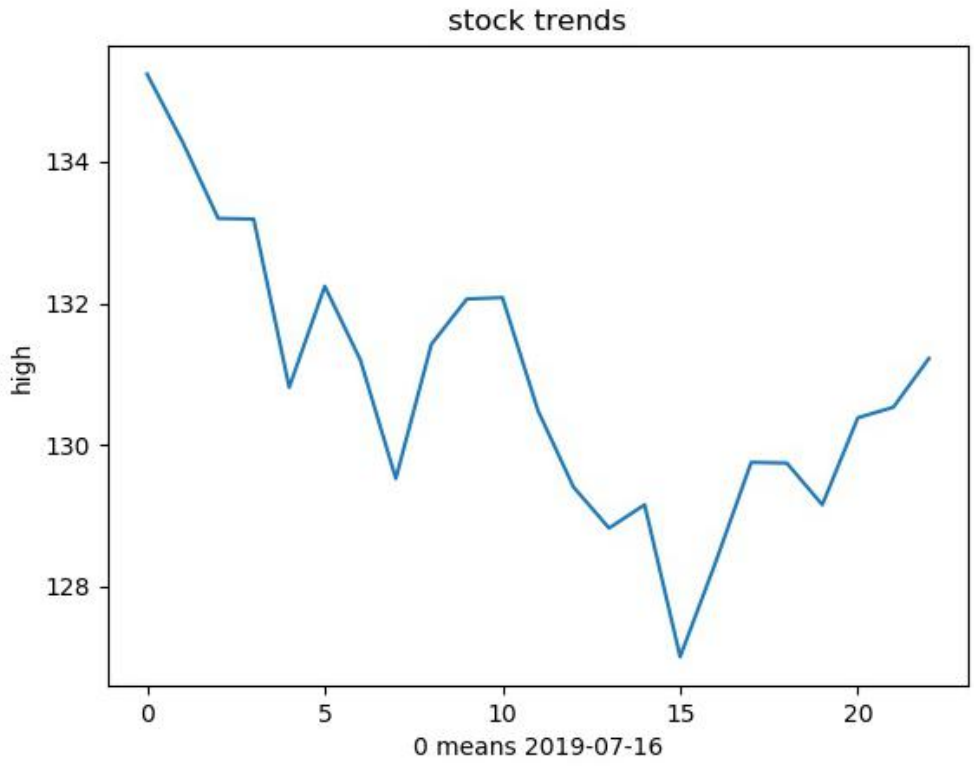


4.10 多轮多次问询穿插其他信息且输入日期错误时系统的处理结果



4. 11 请求趋势图以图片形式回复





4. 11 语音类信息的展示

五、心得体会

通过此次项目，我大大提升了 python 编程的代码功底，在以往的 python 编程经历中，我一直都是处于一种心浮气躁的状态，在调用库的时候从来不去细致的研究它，经常导致调用失败，又不知道问题出在哪里。

此次远程项目，每节课张老师在授课时都耐心的讲解需要用到的代码，讲解的非常细致到位，而且都是等确认每位学生都彻底理解吃透后再进行下一步的教学，这种沉下心来细致研究的态度改变了我对于构建工程时的态度和看法，只有把所做的东西中的每一部分，每一模块都研究透彻了，后面才会少犯错，换句话说，在写代码时沉下心来研究你所要调用的库函数的类并不会浪费你多少时间，它反而会为你在后面编程时省去很多调错的时间，这种感受，让我在本项目的实践过程中感受强烈。在将代码集成在 telegram 的过程中，由于 telegram 的开发文档纯英文，又臭又长，关键部分不详细，检索方式反人类，因此在初期我非常的烦躁，最后慢慢沉下心来摸索着，从代码中的构造类声明文件里跳来跳去，并仔细研究 telegram 的源码，以及其注释，终于把我所需要用的所有功能搞清楚了。

对于本项目，由于之前曾经有过将图灵机器人运用到嵌入式系统的开发经历，因此我对于聊天机器人的构建原理以及方法一直有着巨大的好奇心，一直想做一个属于自己的 bot，这个项目完全满足了我的想法，我收获非常丰厚，尤其是张老师最后一节课说的“市面上即便是 siri 这样精良的机器人，运用的也都是我给你们讲解的这些技术，只不过它运用的更好”，让我感到非常的有成就感。

此外，在教学过程中，张老师一直在强调着“不要迷信深度学习”也让我有所警醒，对于本项目，确实，虽然使用的方法并不是深度学习，但是我发现，确实单单依靠训练好的模型去预测，真的效果有时候并没有一些传统的，看似僵硬的方法来的好，二者相互结合，取长补短，才能获得较好的效果。