

Design of Chat-bot in finance area

Tong Liu

Introduction

Nowadays, with the rapid development of computer technologies and artificial intelligence (AI). Chat-bots are used in many fields and in everyone's life. Chat-bot is a program that can understand the user's statement (in natural language) and give responses to the users just like a human. Some of these chat-bots are complex, which can understand most of the user's statement, and deal with a lot of different things, such as the smart assistant apps in smart phones. Some of them are simple and can only understand some of the intents related to some specific field of the user and can only handle things in its own area. But most of them are based on the same technologies and principles. Such design is a Chat-Bot that can handle some of the finance stock-related problems, such as stock price query, candlestick graph querying. It can also query company information.

Technologies Behind

Rasa NLU

Rasa_NLU is a powerful Natural Language Understanding (NLU) framework in python. It does not have NLU algorithms included in its own package, but external components like Spacy. It mainly uses machine learning algorithms in Spacy to process the sentences, but it also supports regular expressions. Its functions include tokenizer, extractor, featurizer and classifier. It can achieve almost all required tasks of NLU. Since it's based on Spacy, it supports different languages. If you want it to understand another language, download a new model. The training data used can be a single file or a series of files in two formats, JSON and Markdown. In this project, markdown will be used because it can be expressed in a simpler way.

Matplotlib

The Matplotlib is a python library that can draw different types of mathematical figures like bar charts, line charts, pie charts and so on. With its plugin 'mpl_finance', it can plot financial figures like candlestick graphs. This library is used to plot candlestick graphs of specified stock when the user asked in this project.

Technological Details

The NLU Training data

Every machine learning framework is needed to be trained before it can be put into use. The more the training data is, the more accurate will the results be in the production stage. In this project, since machine learning is used by NLU framework when parsing the user's input, it should also be trained. The training data is consist

of many example sentences with the entities and the intent marked. The framework is trained every time once the program has started. The training data in either JSON format or Markdown format are accepted by the framework, but the Markdown format is more convenient to compose. The project uses Markdown format, part of the data with same intent is shown below.

```
## intent:company_info_query
- company information
- can you tell me the company information
- i want the information about [Apple Inc.](company_name)
- i want to know what [Alphabit Inc.](company_name) is
- who is the [CEO](company_info) of [Bilibili Inc.](company_name)
- the [CEO](company_info) of [Bilibili Inc.](company_name)
- [webistel](company_info) of [Bilibili Inc.](company_name)
- where is [Bilibili Inc.](company_name)
- what's the [website](company_info) of [Bilibili Inc.](company_name)
- can you tell me the [address](company_info) of [Bilibili Inc.](company_info)
- what's the name of [CEO](company_info) of [BILI](stock_id)
```

Fig 1: part of the training data

The training data is not very large in amount in this project, so the result might be odd sometimes.

NLU Process

When the user data is passed to the bot, we have to understand them. The `parse()` method of the interpreter will be called and the intent and the entities of the user's statement is extracted. After that, the confidence of the intent will be evaluated. If the confidence is too low (below 0.3), we can think that the bot does not understand the user's statement because at this occasion the user would said something unrelated. In this occasion, a message showing that the bot don't understand the meaning of the statement instead of showing wrong results. The entities that can be seen as parameters for the query is stored in a dictionary, which will be cleared after each 'query session'.

There can be more than one round of dialog in one 'query session'. When the user shows the intent of querying something but some necessary condition of the query is missing. There are two state variables in the bot. One showing which parameter is going to be asked from the user, and the other shows what is the intent of the last statement. Once the bot have entered the state of asking the user for some information, only the information asked will be accepted, otherwise the bot will send a message to require the user to input again. Once the required parameters is obtained, it will be stored into the parameters dictionary and the query will continue. Such state can be reset to the initial state anytime if your statement shows the intent of 'reset'. Once the state have reset, the two state variables will be reset and the parameters dictionary will be cleared, then the user can ask the bot other questions.

Once the required parameters is obtained, the data will be passed to the corresponding query function and the result will be returned.

Once there are some statement that should be returned to user, the data are formatted to a user-friendly sentence and the sentence will be shown to user.

The following code shows the variables of the Bot class, which is the class that handles all the user's input:

```

class Bot:
    def __init__(self):
        trainer = Trainer(config.load('spacy_config.yaml'))
        train_data = load_data('training.md')
        self.interpreter = trainer.train(train_data) # The natural language interpreter trained from the training data
        self.state = "init" # The state of the bot, can be reset anytime
        self.last_intent = "none" # Previous intent, useful for multi-time dialog, this is another state variable
        self.query = IEXQuery() # The IEX cloud query object
        self.stock = "" # Current stock id, avoiding setting same stock id for more than once
        self.params = {} # Stored params for multi-time dialog

```

Fig 2. Variables in Bot class

Data Querying

All the stock related data is obtained from IEX Cloud. IEX Cloud is a platform that makes financial data and services accessible to everyone. Such platform provides API for developers to access finance data. IEX Cloud have free and paid services, but in this project only free data are used. Each user, no matter free or paid, requires a API token to access the data. The API token is generated when the user is registered.

In this project we use a python library, iexfinance, to access the data from IEX Cloud. With such library, we don't need to handle HTTP requests and responses of the IEX Cloud platform in our own code. The querying functions are coded in a class called IEXQuery.

Candlestick graphs

This bot can obtain a series of historical stock price data and plot a candlestick graph for the stock in a specific date range. The process of plotting the graph is achieved by the matplotlib library and its mpl_finance plugin. The obtained data should be simplified because if too many data is passed into the graph, the process of plotting could be very slow and the graph could be unreadable.

The following code shows the process of plotting the graph:

```

# Show the candlestick graph of the current stock from a specified date to now
def plot_graph(self, start):
    price_sheet = None
    try:
        # obtain the data from IEX cloud. we use json format since it's easy to process
        # chartSimplify meant to simply the data, since we don't want to many data to plot the graph
        price_sheet = get_historical_data(self.symbol_info['symbol'], start, str(datetime.date.today()),
                                          output_format='json', chartSimplify=True, token=API_KEY)
    except IEXQueryError as ex:
        return "Sorry, but I have trouble obtaining the balance data."

    try:
        # Import the Matplotlib and Matplotlib Finance libraries
        import matplotlib.pyplot as plt
        import mpl_finance as mpf
        from matplotlib.pylab import date2num

        # Convert the obtained data to the format that Matplotlib can understand
        data = []
        for k in price_sheet.keys():
            a = price_sheet[k]
            # Convert the dates to float, making Matplotlib understand them
            date_time = datetime.datetime.strptime(k, '%Y-%m-%d')
            data.append([date2num(date_time), a['open'], a['close'], a['high'], a['low'], a['volume']])

        fig, ax = plt.subplots(figsize=(1200 / 72, 480 / 72))
        fig.subplots_adjust(bottom=0.1)
        mpf.candlestick_ohlc(ax, data, colordown='#53c156', colorup='#ff1717', width=0.3, alpha=1)
        ax.grid(True)
        ax.xaxis_date()

        # Now return the figure instead of string.
        # Different user interfaces have different ways to show figures
        return fig
    except Exception as e:
        # Once there's errors, we just return a error message.
        return "Oops, something wrong happened while plotting the graph!"

```

Figure 3. Code for plotting the graph

User Interfaces

1. **Command-line interface:** The interface that use the system console for interaction with users. The user types directly into the console, and the bot print its response in the console. If the bot returned a list containing multiple strings, it will be regard as multiple responses. If the bot returned a figure, the figure will be shown in a separate window.

To start it, run the cli.py script

2. **WeChat interface:** This interface use popular IM app WeChat for interaction with users. When started, the program will first ask the user to scan a QR code with WeChat app to login to WeChat, then enter the name of friend you want the bot to chat with. The functions are provided by the wxpy library.

To start it, run the wechat.py script. NOT TESTED because my account cannot be logged in to WeChat Web.

Testing Results

Testing Environment

Hardware:

Model: ThinkPad x260

CPU: Intel® Core™ i5-6300U CPU
RAM: 8 GB
Disk: 120GB SSD

Software:

OS: Ubuntu 18.10

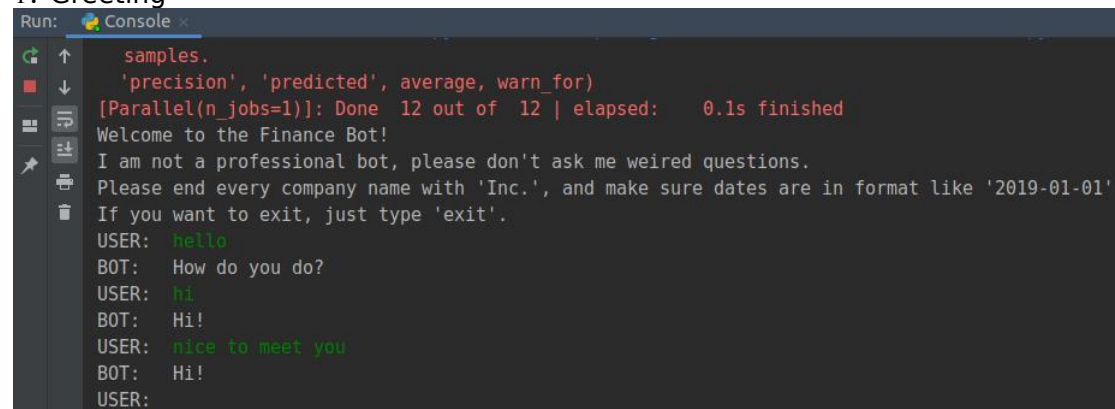
Runtime:

Anaconda 1.9.7

Python 3.6.9

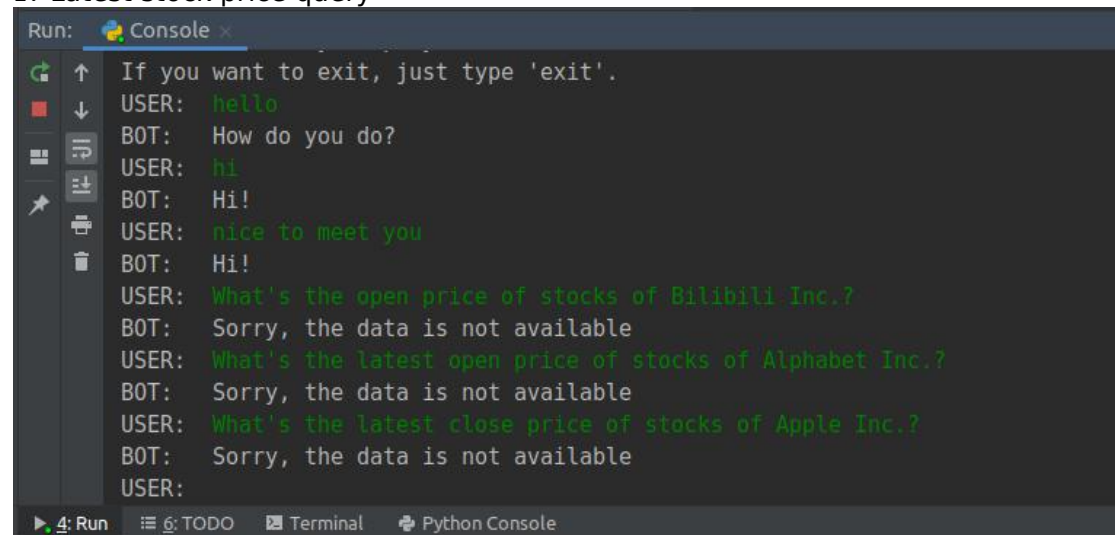
Testing Results

1. Greeting



```
Run: Console x
samples.
'precision', 'predicted', average, warn_for)
[Parallel(n_jobs=1)]: Done 12 out of 12 | elapsed: 0.1s finished
Welcome to the Finance Bot!
I am not a professional bot, please don't ask me weird questions.
Please end every company name with 'Inc.', and make sure dates are in format like '2019-01-01'
If you want to exit, just type 'exit'.
USER: hello
BOT: How do you do?
USER: hi
BOT: Hi!
USER: nice to meet you
BOT: Hi!
USER:
```

2. Latest stock price query



```
Run: Console x
If you want to exit, just type 'exit'.
USER: hello
BOT: How do you do?
USER: hi
BOT: Hi!
USER: nice to meet you
BOT: Hi!
USER: What's the open price of stocks of Bilibili Inc.?
BOT: Sorry, the data is not available
USER: What's the latest open price of stocks of Alphabet Inc.?
BOT: Sorry, the data is not available
USER: What's the latest close price of stocks of Apple Inc.?
BOT: Sorry, the data is not available
USER:
```

(Data unavailable due to no data is present on IEX Cloud)

3. Historical price query


```
Run: Console x
samples.
'precision', 'predicted', average, warn_for)
[Parallel(n_jobs=1)]: Done 12 out of 12 | elapsed: 0.1s finished
Welcome to the Finance Bot!
I am not a professional bot, please don't ask me weird questions.
Please end every company name with 'Inc.', and make sure dates are in format like '2019-01-01'
If you want to exit, just type 'exit'.
USER: What's the open price of stocks of Bilibili Inc. at 2018-10-10?
BOT: The open price of stocks of BILI at 2018-10-10 is 13.26 USD
USER: What's the stock price of Apple Inc. at 2019-01-10?
BOT: Please tell me which type of price you're looking for. Possible price types are: open, close, high, low, and volume
USER: open price, please
BOT: The open price of stocks of AAPL at 2019-01-10 is 152.5 USD
USER:
```

(The second and third dialog shows two rounds of dialog in one query session)

4. Company Information

```
Run: Console x
USER: open price, please
BOT: The open price of stocks of AAPL at 2019-01-10 is 152.5 USD
USER: Tell me something about Bilibili Inc.
BOT: Bilibili, Inc. engages in the provision of content and entertainment experience.
    It covers genres and media formats, including videos, live broadcasting, and mobile
    games. The company was founded by Xu Yi in December 2013 and is headquartered
    Shanghai, China.
USER: Who is the CEO of Bilibili Inc.?
BOT: The CEO of the company is Chen Rui.
USER: can you tell me something about the company?
BOT: You should tell me which company you're querying. Tell me the stock ID or the
    company name (ends with 'Inc.')
```

```
USER: Apple Inc.
BOT: Apple, Inc. engages in designing, manufacturing, and marketing of mobile
    communication, media devices, personal computers, and portable digital music players.
    It operates through the following geographical segments: Americas, Europe, Greater
    China, Japan, and Rest of Asia Pacific. The Americas segment includes North and South
    America. The Europe segment consists of European countries, as well as India, the
    Middle East, and Africa. The Greater China segment comprises of China, Hong Kong, and
    Taiwan. The Rest of Asia Pacific segment includes Australia and Asian countries. Its
    products and services include iPhone, iPad, Mac, iPod, Apple Watch, Apple TV, a
    portfolio of consumer and professional software applications, iPhone OS (iOS), OS X
    and watchOS operating systems, iCloud, Apple Pay and a range of accessory, service
    and support offerings. Apple was founded by Steven Paul Jobs, Ronald Gerald Wayne,
    and Stephen G. Wozniak on April 1, 1976 and is headquartered in Cupertino, CA.
USER:
```

(The last two rounds of dialog shows two rounds of dialog in one query session)

5. Candlestick graph

Console output:

```
Run: Console x
I am not a professional bot, please don't ask me weird questions.
Please end every company name with 'Inc.', and make sure dates are in format like '2019-01-01'
If you want to exit, just type 'exit'.
USER: plot the graph of Alphabet Inc.
BOT: Which day do you want the candlestick graph start from?
USER: 2018-01-01
BOT: Here is the graph showing the price and volume variation of GOOG in past 2018 - 01 - 01.
BOT: [The bot sent a graph and it's shown in a separate window.]
/home/lt/anaconda3/envs/ML/lib/python3.6/site-packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning:
    plotting method. The converter was registered by pandas on import. Future versions of pandas will require
    To register the converters:
    >>> from pandas.plotting import register_matplotlib_converters
    >>> register_matplotlib_converters()
    warnings.warn(msg, FutureWarning)
USER: |
```

Graph Window:



6. Resetting state

```
Run: Console x
↑ 'precision', 'predicted', average, warn_for)
↓ [Parallel(n_jobs=1)]: Done 12 out of 12 | elapsed: 0.1s finished
Welcome to the Finance Bot!
I am not a professional bot, please don't ask me weird questions.
Please end every company name with 'Inc.', and make sure dates are in format like '2019-01-01'
If you want to exit, just type 'exit'.
USER: can you tell me the price of stocks of Apple Inc.
BOT: Please tell me which type of price you're looking for. Possible price types are: open, close, high, low
USER: cancel operation
BOT: Ok, let's do other things first, previous cancelled.
USER: tell me something about Alphabet Inc.
BOT: Alphabet, Inc. is a holding company, which engages in the business of acquisition and operation of
Google segment includes its main Internet products such as Ads, Android, Chrome, Commerce, Google
includes businesses such as Access, Calico, CapitalG, GV, Nest, Verily, Waymo, and X. The company is
headquartered in Mountain View, CA.
USER:
```

(After the first round of dialog, the bot is in state to wait for user's answer about the price type. The state is reset after the user says 'cancel operation')