Stocktwits collector

Release 0.3.0

Alessandra Bilardi

Mar 12, 2023

CONTENTS:

This package contains the class for collecting the twits of Stocktwits on your local.

CHAPTER

ONE

GETTING STARTED

Stocktwits collector package is implemented for collecting the twits of Stocktwits on your local.

The goal is to implement this package for each Stocktwits API and to manage how to download many data on more files to avoid networking issues and to require again all data but only the missing part.

It is part of the educational repositories to learn how to write stardard code and common uses of the TDD.

1.1 Installation

If you want to use this package into your code, you can install by python3-pip:

```
pip3 install stocktwits_collector
python3
>>> import stocktwits_collector.collector as Collector
>>> help(Collector)
```

1.2 Development

The package is not self-consistent. So after to have downloaded the package by github and you have to install the requirements:

```
git clone https://github.com/bilardi/stocktwits-collector
cd stocktwits-collector/
pip3 install --upgrade -r requirements.txt
```

See the documentation to contribute.

1.3 Documentation

Read the documentation on readthedocs for

- Usage
- Development

1.4 Change Log

See CHANGELOG.md for details.

1.5 License

This package is released under the MIT license. See LICENSE for details.

CHAPTER

TWO

API

2.1 Collector

The class for collecting twits of Stocktwits

A collection of methods to simplify your downloading

2.1.1 Stocktwits Collector

stocktwits_collector.collector.Collector.get_history	get history from Stocktwist, default last 30 messages
stocktwits_collector.collector.Collector.save_history	save history from Stocktwist on files splitted by chunk per day, week or month

2.1.2 Detailed list

class stocktwits_collector.collector.**Collector**

clean_data(*messages*, *event*)

clean data

Arguments:

messages (list of dict)
list of messages

event (dict)
dictionary fully described in save_history() symbols (list of str): names of symbols to fetch users (list of str): names of users to fetch only_combo (bool): if True, fetches only messages of those symbols posted from those users

Returns:
list of unique dictionaries cleaned

clean_history(*cursor*, *history*, *chunk='day'*)

clean history from messages with different chunk

Arguments:

cursor (dict)
dictionary with the keys oldest_date, min ID, earliest_date and max (ID)

history (list[dict])
list of messages

chunk (str)
day, week or month, default day

Returns:
history cleaned

`get_cursor`(*messages*)

get cursor with oldest date, min ID and max ID

Arguments:

messages (list[dict])
list of messages

Returns:
a dictionary with oldest_date, min ID, earliest_date and max (ID)

`get_data`(*event*)

get data from Stocktwits, default last 30 messages

Arguments:

event (dict)
dictionary fully described in save_history() symbols (list of str): names of symbols to fetch users (list of str): names of users to fetch min (int): optional, min ID max (int): optional, max ID limit (int): optional, defalt 30 messages

Returns:
list of messages

`get_date`(*chunk='day'*, *date=None*, *jump_chunk=False*)

get date at midnight about chunk

Arguments:

chunk (str)
day, week or month, default day

date (str)
datetime with format %Y-%m-%dT%H:%M:%SZ

jump_chunk (bool)
True if you want to jump one chunk

Returns:
string of date at midnight about that chunk or next one

`get_file_name`(*history*, *current_chunk*, *event*)

get filename

Arguments:

history (list[dict])
list of messages

current_chunk (dict)
dictionary like event

event (dict)
dictionary fully described in save_history()

Returns:
the file name

`get_history`(*event*)

get history from Stocktwist, default last 30 messages

Arguments:

event (dict)
dictionary fully described in save_history() start (datetime): optional, min datetime is_verbose (bool): optional, if True comments will be printed

Returns:
list of messages

`get_temporary_event`(*messages*, *current_chunk*, *event*)

get temporary chunk event from messages

Arguments:

messages (list[dict])
list of messages

current_chunk (dict)
dictionary fully described in save_history()

event (dict)
dictionary fully described in save_history()

Returns:
the temporary chunk event updated with the partial start and new min

`hold_output`()

hold output

This method is temporary until PR approval: https://github.com/p-hiroshige/stockTwitsAPI/pull/1

Example:

with hold_output() as (out, err):
method_with_a_print()

captured_output = out.getvalue().strip()

`is_same_chunk`(*first_date*, *second_date*, *chunk='day'*)

compare a date with a second date

Argument:

first_date (str)
datetime with format %Y-%m-%dT%H:%M:%SZ

second_date (str)
another date with format %Y-%m-%dT%H:%M:%SZ

chunk (str)
day, week or month, default day

Returns:
a boolean, True if the dates are of the same chunk

`is_younger`(*first_date*, *second_date*)

compare a date with a second date

Argument:

first_date (str)
datetime with format %Y-%m-%dT%H:%M:%SZ

second_date (str)
another date with format %Y-%m-%dT%H:%M:%SZ

Returns:
a boolean, True if first date is younger than second one

`save_data`(*history*, *current_chunk*, *event*)

save data

Arguments:

history (list[dict])
list of messages

current_chunk (dict)
dictionary like event

event (dict)
dictionary fully described in save_history()

Returns:
the temporary chunk event updated with the partial start and new max

`save_history`(*event*)

save history from Stocktwist on files splitted by chunk per day, week or month

Arguments:

event (dict)
symbols (list[str]): names of symbols to fetch users (list[str]): names of users to fetch only_combo (bool): optional, if True, fetches only messages of those symbols posted from those users min (int): optional, min ID max (int): optional, max ID limit (int): optional, default 30 messages start (str): optional, min datetime chunk (str): optional (day, week or month), default day filename_prefix (str): optional, default "history." filename_suffix (str): optional, default ".json" is_verbose (bool): optional, if True comments will be printed

Returns:
last temporary chunk event discarded

`there_is_symbol`(*symbols_fetched*, *symbols_target*)

check if in the message there are the symbols target

Arguments:

symbols_fetched (list of dict)
list of symbols

symbols_target (list of string)
list of symbols names

Returns:
a boolean, True if there is at least one symbol of target in the symbols fetched

`update_event`(*key*, *value*, *event*)

update a specific key of event

Arguments:

key (str)
attribute name of event

value (mix)
value you want to replace on that key

event (dict)
dictionary fully described in save_history()

Returns:
dictionary with the attribute named key changed with value

`walk`(*event*, *cursor*, *history*)

walk along the messages like a shrimp

Arguments:

event (dict)
dictionary fully described in save_history()

cursor (dict)
dictionary with the keys oldest_date, min ID, earliest_date and max (ID)

history (list[dict])
list of messages

Returns:
cursor, history

CHAPTER

THREE

USAGE

The package uses the Stocktwits API manages three type of streas: user, symbol and conversation. Now the package manages the user and symbol streams.

There are some parameters that you can use. These are the mandatory parameters:

- **symbols**, you can define a list of symbols that you want to download: this list has to have at least one element or it has to exist the parameter **users**
- **users**, you can define a list of users that you want to download: this list has to have at least one element or it has to exist the parameter **symbols**

And these are optionals:

- **only_combo**, when you want to download only the combo between a specific symbol and user, you have to use each previous parameter and this that it is a boolean
- **min**, it is the ID of a specific twit from which you want to start downloading
- **max**, it is the ID of a specific twit where you want to stop downloading
- **limit**, it is the number of messages that you want to download in one shot
- **start**, it is the datetime from which you want to start downloading
- **chunk**, it is the chunk (day, week or month) in which you want to split the data
- **filename_prefix**, it is the prefix name of files where you want to save the data
- **filename_suffix**, it is the suffix name of files where you want to save the data
- **is_verbose**, when you want to print some information to understand what the system is saving, it is a boolean

Without optional parameters, the system downloads the last 30 messages and prints those in the output. If you want to save that on a file (or more files), you have to use at least the **chunk** parameter.

3.1 Examples

Remeber to install the package by pip

```
pip3 install stocktwits-collector
```

or by requirements.txt contains one line with **stocktwits-collector**

```
pip3 install --upgrade -r requirements.txt
```

```
import os
import json
import pandas as pd

from stocktwits_collector.collector import Collector
sc = Collector()

# download last messages up to 30
messages = sc.get_history({'symbols': ['TSLA'], 'limit': 4})
# download the messages from a date to today
messages = sc.get_history({'symbols': ['TSLA'], 'start': '2022-04-04T00:00:00Z'})
# save the messages on files splitted per chunk from a date to max ID
chunk = sc.save_history({'symbols': ['TSLA'], 'start': '2022-04-04T00:00:00Z', 'chunk':
↪'day'})

# load data from one file
with open('history.20220404.json', 'r') as f:
    data = json.loads(f.read())
df = pd.json_normalize(
    data,
    meta=[
        'id', 'body', 'created_at',
        ['user', 'id'],
        ['user', 'username'],
        ['entities', 'sentiment', 'basic']
    ]
)
twits = df[['id', 'body', 'created_at', 'user.username', 'entities.sentiment.basic']]

# load data from multiple files
frames = []
path = '.'
for file in os.listdir(path):
    filename = f"{path}/{file}"
    with open(filename, 'r') as f:
        data = json.loads(f.read())
        frames.append(pd.json_normalize(
            data,
            meta=[
                'id', 'body', 'created_at',
                ['user', 'id'],
                ['user', 'username'],
                ['entities', 'sentiment', 'basic']
            ]
          )
        )
df = pd.concat(frames).sort_values(by=['id'])
twits = df[['id', 'body', 'created_at', 'user.username', 'entities.sentiment.basic']]
```

CHAPTER

FOUR

DEVELOPMENT

The package uses the Stocktwits API manages three type of streas: user, symbol and conversation. Now the package manages the user and symbol streams.

You can contribute to implement other functionalities by a Pull Request to master branch.

4.1 Run tests

```
cd stocktwits-collector/
pip3 install --upgrade -r requirements.txt
python3 -m unittest discover -v
```

There is also a script for integration tests, but it is only for specific changes

```
# run API with chunk day # around 160s
python3 -m unittest tests/integration_test.py
# run API with chunk week # around 400s
CHUNKS=week python3 -m unittest tests/integration_test.py
# run API with chunk month # around 1600s
CHUNKS=month python3 -m unittest tests/integration_test.py
# run API with chunk day, week and month
CHUNKS=all python3 -m unittest tests/integration_test.py
# run API with verbose and chunk day
VERBOSE=True python3 -m unittest tests/integration_test.py
```

4.2 Run make

Makefile is useful for many actions:

- run the unit test by `make unittest`
- run the doc build by `make doc`

4.3 Prepare a Pull Request (PR)

You can fork the repository in your space and then you can clone your copy in your local to change and run tests.

```
cd stocktwits-collector/
pip3 install --upgrade -r requirements.txt
python3 -m unittest discover -v
git checkout -b your-branch
git add files-changed
git commit -m "describe your changes here"
git push origin push your-branch
```

You can create the PR from your fork.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX