Telegram-API: a Python-based open-source tool for Telegram

Overview

This tool connects to Telegram's API. It generates JSON files containing channel's data, including channel's information and posts. You can search for a specific channel, or a set of channels provided in a text file (one channel per line.)

Files are saved by default in a folder called *output/data*. These folders are created by the script. You can also give a specific output directory to store collected data.

Software required

- Python 3.x

- Telegram API credentials
    - Telegram account
    - App `api_id`
    - App `api_hash`

Python required libraries

- Telethon
- Pandas
- Openpyxl
- tqdm
- Networkx
- Matplotlib
- Louvain Community Detection

Installing

- **Via git clone**

```
git clone https://github.com/estebanpdl/telegram-api.git
```

This will create a directory called `telegram-tracker` which contains the Python scripts. Cloning allows you to easily upgrade and switch between available releases.

- **From the github download button**

Download the ZIP file from github and use your favorite zip utility to unpack the file `telegram-tracker.zip` on your preferred location.

After cloning or downloding the repository, install the libraries from `requirements.txt`.

```
pip install -r requirements.txt
```

or

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

Once you obtain an API ID and API hash on my.telegram.org, populate the `config/config.ini` file with the described values.

```
[Telegram API credentials]
api_id = api_id
api_hash = api_hash
phone = phone
```

Note: Your phone must be included to authenticate for the first time. Use the format + <code><number> (e.g., +19876543210). Telegram API will send you a code via Telegram app that you will need to include.

Example usage

`main.py`

This Python script will connect to Telegram's API and handle your API request.

Options

- `--telegram-channel` Specifies Telegram Channel to download data from.
- `--batch-file` File containing Telegram Channels to download data from, one channel per line.
- `--limit-download-to-channel-metadata` Will collect channels metadata only, not channel's messages. (default = False)
- `--output, -o` Specifies a folder to save collected data. If not given, script will generate a default folder called `./output/data`
- `--min-id` Specifies the offset id. This will update Telegram data with new posts.

Structure of output data

```
├──📁 output
|   └──📁 data
|       └──📁 <channel_name>
|               └──<channel_name>.json
|               └──<channel_name>_messages.json
|       └──chats.txt // TM channels, groups, or users' IDs found in data.
|       └──collected_chats.csv // TM channels or groups found in data (e.g.,
forwards)
|       └──collected_chats.xlsx // TM channels or groups found in data
(e.g., forwards)
|       └──counter.csv // TM channels, groups or users found in data (e.g.,
forwards)
|       └──user_exceptions.txt // From collected_chats, these are mostly TM
users' which
|
metadata was not possible to retrieve via the API
|       └──msgs_dataset.csv // Posts and messages from the requested
channels
```

Examples

Basic request

```
python main.py --telegram-channel channelname`
```

Expected output

- Files of collected channels:
  - chats.txt
  - collected_chats.csv
  - user_exceptions.txt
  - counter.csv
- A new folder: <*channel_name*> containing
  - A JSON file containing channel's profile metadata

- A JSON file containing posts from the requested channel

Request using a text file containing a set of channels

```
python main.py --batch-file './path/to/channels_text_file.txt'
```

Expected output

- Files of collected channels:
  - chats.txt
  - collected_chats.csv
  - user_exceptions.txt
  - counter.csv
- New folders - based on the number of requested channels: <*channel_name*> containing
  - A JSON file containing channel's profile metadata
  - A JSON file containing posts from the requested channel

These examples will retrieve all posts available through the API from the requested channel. If you want to collect channel's information only, without posts, you can run:

Limit download to channel's metadata only

```
python main.py --telegram-channel channelname --limit-download-to-channel-metadata
```

or, using a set of telegram channels via a text file:

```
python main.py --batch-file './path/to/channels_text_file.txt' --limit-download-to-channel-metadata
```

Updating channel's data

If you want to collect new messages from one channel, you need to identify the message ID from the last post. Once you identify the id, run:

```
python main.py --telegram-channel channelname --min-id 12345
```

Expected output

- Files of collected channels:
  - chats.txt
  - collected_chats.csv
  - user_exceptions.txt
  - counter.csv
- A new folder: <*channel_name*> containing
  - A JSON file containing channel's profile metadata
  - A JSON file containing posts from the requested channel

Specify output folder

The script allows you to specify a specific output directory to save collected data. The sxcript will create those folders in case do not exist.

```
python main.py --telegram-channel channelname --output
./path/to/chosen/directory`
```

The expected output is the same a described above but data will be save using the chosen directory.

---

build-datasets.py

This Python script reads the collected files and creates a new dataset containing messages from the requested channels. By default, the created dataset will be located in the `output` folder.

If you provided a specific directory to save collected data, you need to provide the same path to use this script.

Options

- `--data-path` Path were data is located. Will use `./output/data` if not given.

If a specific directory was not provided in `main.py`, run:

```
python build-datasets.py
```

If you provided a specific directory using the option `--output` in `main.py`, run:

```
python build-datasets.py --data-path ./path/to/chosen/directory
```

These option will create the above-mentioned dataset: `msgs_dataset.csv`, a file containing posts and messages from the requested channels.

---

channels-to-network.py

This Python script builds a network graph. By default, the file will be located in the `output` folder. The script also saves a preliminary graph: `network.png` using the modules matplotlib, networkx, and python-louvain, which implements community detection. You can import the GEFX Graph File using different softwares, including Gephi.

Options

- `--data-path` Path were data is located. Will use `./output/data` if not given.

If a specific directory was not provided in `main.py`, run:

```
python channels-to-network.py
```

If you provided a specific directory using the option `--output` in `main.py`, run:

```
python channels-to-network.py --data-path ./path/to/chosen/directory
```