SMS Scraper

python version: 3.10

Get SMS from page: receive-smss.com

Workflow

The program get all links / phone numbers from home page, open each phone page with multithreading, extract each received message and send data to the API.

Multithreading

The program get the number from the home page with a single thread, after it, extract the messages with multithreading. You can configure the number of thread to run at the same time (more details in *Settings* section).

Duplicated messages

For avoid duplicated, the program use a a data base. You can create the table with the script: **sql/create_table.sql**

For each row in the SMS table, from each number, the script check if the current message have been saved in the the table, if yes, it means that all the message after the current message, have already been extracted and sent to the API, so the programa skips them an continues with the next table.

Note: The first time you run the program, it will take several minutes because it must extract all messages from all numbers. Later the data extraction will be much faster.

Install

Third party modules

Install all modules from pip:

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

Run

For run the program, run the **__ main__.py** file or the **project folder**, with your python 3.10 interpreter.

Settings

config.json

All setting are saved in the **config.json** file.

```
{
    "threads_num": 50,
    "debug_mode": false,
    "loop_mode": true,
    "wait_time": 60,
    "api_key": "p8AQEUBBW**********",
    "dbname": "sms",
    "table": "history",
    "user": "daridev2",
    "password": "alice1999++",
    "hostname": "localhost"
}
```

- ### threads_num

Number of pages (threads) to requests data at the same time.

Try different number for found you best settings.

Note: Using a low number will be the program slower, but it will consume less resources. Use a lot of threads will consum more resources, but the program will be faster.

- ### debug_mode

true o **false**. If *true*, the program extract the data, but dont made any api Call, if *false*, send all data to the api.

*Note: this variable its usefull for testing (for example, for found the best number of threads). but if you run in any way that hides the terminal (like cron), use this option should be in **false***

- ### loop_mode

If **true**, run the web scraping in infinity loop. If **false** only run one time and after end the program.

- ### wait_time

Minimum of **seconds** to wait after each web scraping loop (only available if *loop_mode* its active).

For sample, if the program takes 40 seconds to extract the new message, and you setup 60 to this variable, the program will wait 20 extra seconds, so that there is at least 60 seconds between each web scraping loop.

Note: this variable is usefull for save resources and avoit web scraping detection.

- ### api_key

Api key for send datas to the API

- ### dbname

Name of the databse with the history table

- ### table

Name of the history table

- ### user

User name for the database

- ### password

User password for the database

- ### hostname

IP / hostname of rthe database (local or remote)

Clean files

Aditional to the *config.json*, the project use some extra local files to save data. If the files are too big, you should clean the contend keeping the basic structure

- ### history.json

File where the messages are saved for avoid duplicates.

Note: if you clean this file, the next time that you run the program it will requiere more time for extract all messages from all phone (if you are traying to found the best threads number, you should clean this file before each test)

basic structure

If you clean the file, keep the following content:

```
{
    "history": [],
    "ids": []
}
```

- ### .log

File for save all debug rows. Usefull for detect errors and see the program flow without terminal (for exaple, when you run the script with cron).

basic structure

This files dosent have a requied structure. You can delete all lines.