whatstk

lucasrodes

Apr 28, 2021

CONTENTS

whatstk is a python package providing tools to parse, analyze and visualize WhatsApp chats developed under the sociepy project. Easily convert your chats to csv or simply visualize statistics using the python library. The package uses pandas to process the data and plotly to visualise it.

The project is distributed under the GPL-3.0 license and is available on GitHub.

CHAPTER

ONE

FIRST CONTACT WITH WHATSTK

whatstk is built around *BaseChat* object interface, which requires class method *from_source* to be implemented. This method loads and parses the source chat file into a pandas.DataFrame.

Below, we use the WhatsApp implementation, i.e. `WhatsAppChat`, to load LOREM chat. To test it with your own chat, simply export it as a txt file to your computer and then use class argument `filepath`, as shown in the following example.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> chat.df.head(5)
                 date        username                                                ␣
↪message
0 2020-01-15 02:22:56            Mary                     Nostrud exercitation magna␣
↪id.
1 2020-01-15 03:33:01            Mary     Non elit irure irure pariatur exercitation.
2 2020-01-15 04:18:42  +1 123 456 789  Exercitation esse lorem reprehenderit ut ex ve.
↪..
3 2020-01-15 06:05:14        Giuseppe  Aliquip dolor reprehenderit voluptate dolore e.
↪..
4 2020-01-15 06:56:00            Mary              Ullamco duis et commodo␣
↪exercitation.
```

CHAPTER

TWO

INSTALLATION & COMPATIBILITY

This project is on PyPI, install it with pip:

```
pip install whatstk
```

Project has been tested with Python 3.7-3.8.

2.1 From source

Clone the project from the official repository

```
git clone https://github.com/lucasrodes/whatstk.git
```

and install it locally

```
cd whatstk
pip install .
```

2.2 Develop

You can also install the version in development directly from github develop branch.

```
pip install
pip install git+https://github.com/lucasrodes/whatstk.git@develop
```

CHAPTER

THREE

SUPPORT

You can ask questions and join the development discussion on Gitter. Use the GitHub issues section to report bugs or request features. You can also check the project roadmap.

For more details, refer to the *contribute section*.

CHAPTER

FOUR

WHY THIS NAME, WHATSTK?

whatstk stands for “WhatsApp Toolkit”, since the project was initially conceived as a python library to read and process WhatsApp chats.

CHAPTER

FIVE

CONTENT:

5.1 About whatstk

whatstk is a python package providing tools to parse, analyze and visualize WhatsApp chats developed under the sociepy project. Easily convert your chats to csv or simply visualize statistics using the python library. The package uses pandas to process the data and plotly to visualise it.

The project is distributed under the GPL-3.0 license and is available on GitHub.

5.1.1 First contact with whatstk

whatstk is built around *BaseChat* object interface, which requires class method *from_source* to be implemented. This method loads and parses the source chat file into a pandas.DataFrame.

Below, we use the WhatsApp implementation, i.e. `WhatsAppChat`, to load LOREM chat. To test it with your own chat, simply export it as a txt file to your computer and then use class argument `filepath`, as shown in the following example.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> chat.df.head(5)
                 date        username                                                 
↪message
0 2020-01-15 02:22:56            Mary                      Nostrud exercitation magna␣
↪id.
1 2020-01-15 03:33:01            Mary     Non elit irure irure pariatur exercitation.
2 2020-01-15 04:18:42  +1 123 456 789  Exercitation esse lorem reprehenderit ut ex ve.
↪..
3 2020-01-15 06:05:14        Giuseppe  Aliquip dolor reprehenderit voluptate dolore e.
↪..
4 2020-01-15 06:56:00            Mary              Ullamco duis et commodo␣
↪exercitation.
```

5.1.2 Installation & compatibility

This project is on PyPI, install it with pip:

```
pip install whatstk
```

Project has been tested with Python 3.7-3.8.

From source

Clone the project from the official repository

```
git clone https://github.com/lucasrodes/whatstk.git
```

and install it locally

```
cd whatstk
pip install .
```

Develop

You can also install the version in development directly from github develop branch.

```
pip install
pip install git+https://github.com/lucasrodes/whatstk.git@develop
```

5.1.3 Support

You can ask questions and join the development discussion on Gitter. Use the GitHub issues section to report bugs or request features. You can also check the project roadmap.

For more details, refer to the *contribute section*.

5.1.4 Why this name, whatstk?

whatstk stands for "WhatsApp Toolkit", since the project was initially conceived as a python library to read and process WhatsApp chats.

5.2 Getting started

Getting started with the library is fairly easy.

5.2.1 Export a WhatsApp chat

Exporting a WhatsApp chat can be easily done from your Android or iOS device. It is done on a chat basis, so if you want to export several chats you will have to export them individually. **When exporting, make sure to select the chats Without Media option**. Once generated, you can send it via mail, so you can save it in your computer.

Android

The export on Android might include several files. We are only interested in the text file (i.e. `txt` extension file).

Fig. 1: Android 9, WhatsApp v2.20.123

For more details, refer to official website.

iOS

The chat is exported as a zip, which can be easily unzipped in your computer.

Fig. 2: iOS 12, WhatsApp v2.20.31

5.2.2 Load chat

Once you have *exported* a chat it is time to load it in python.

In this example we load the example LOREM chat, which is available online, using library class `WhatsAppChat`.

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
```

Once loaded, we can check some of the chat messages by accessing its attribute *df*, which is a pandas.DataFrame with columns *username* (name of user sending the message), *message* (message sent) and *date* index (timestamp of message).

```
>>> chat.df.head(5)
                  date        username                                                ␣
↪message
    0 2020-01-15 02:22:56            Mary                     Nostrud exercitation␣
↪magna id.
    1 2020-01-15 03:33:01            Mary     Non elit irure irure pariatur␣
↪exercitation.
    2 2020-01-15 04:18:42  +1 123 456 789  Exercitation esse lorem reprehenderit ut␣
↪ex ve...
    3 2020-01-15 06:05:14        Giuseppe  Aliquip dolor reprehenderit voluptate␣
↪dolore e...
    4 2020-01-15 06:56:00            Mary              Ullamco duis et commodo␣
↪exercitation.
```

Getting the start and end date of the chat can give us a good overview of the chat content.

```
>>> print(f"Start date: {chat.start_date}\nEnd date: {chat.end_date}")
Start date: 2020-01-15 02:22:56
End date: 2020-05-11 22:32:48
```

Also, getting a list with the chat members is simple

```
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

See also:

- *Load chat from multiple sources*
- *Load a chat with specific hformat*

5.2.3 Command line

whatstk provides a set of command line tools to obtain quick results using the command line. To use these, make sure that you have previously *installed the library*.

For instance, convert a WhatsApp text file to a CSV file using

```
whatstk-to-csv [input_filename] [output_filename]
```

For more details, check the *command line tools documentation*.

5.2.4 The header format

In WhatsApp, a chat file syntax can differ between devices, OS and language settings, which makes it hard to correctly parse the data for all formats.

The header appears for each message sent in the chat and contains the timestamp and name of the user that sent the message.

See it for yourself and open *an exported chat file*. You will find that the messages have a similar format like the one below:

```
15.04.2016, 15:04 - You created group "Sample Group"
06.08.2016, 13:18 - Messages you send to this group are now secured with end-to-end␣
↪encryption. Tap for more info.
06.08.2016, 13:23 - Ash Ketchum: Hey guys!
06.08.2016, 13:25 - Brock: Hey Ash, good to have a common group!
06.08.2016, 13:30 - Misty: Hey guys! Long time haven't heard anything from you
06.08.2016, 13:45 - Ash Ketchum: Indeed. I think having a whatsapp group nowadays is␣
↪a good idea
06.08.2016, 14:30 - Misty: Definetly
06.08.2016, 17:25 - Brock: I totally agree
07.08.2016, 11:45 - Prof. Oak: Kids, shall I design a smart poke-ball?
```

In this example, the header is **day.month.year, hour:minutes - username:** which corresponds to the header format (a.k.a. **hformat**) `'%d.%m.%y, %H:%M - %name:'`. However, in your case it may be slightly different depending on your phone settings.

Check the table below to see the codes for each header format unit:

Table 1: header format units

Date unit code	Description
'%y'	Year
'%m'	Month of the year (1-12)
'%d'	Day of the month (1-31)
'%H'	Hour 24h-clock (0-23)
'%P'	Hour 12h-clock (1-12)
'%M'	Minutes (0-60)
'%S'	Seconds (0-60)
'%name'	Name of user

See also:

Loading chat using hformat

5.2.5 Library available chats

For the purpose of showcasing code examples and benchmarking different implementations, we have created a pool of chats, hosted in the official repository page. If you want to test the library with one of your own tests, check in the *code examples*.

The chats are available via their corresponding URLs, which are listed in source code `whatstk.data`.

Contents

WhatsApp

Object `whatsapp_urls` contains all URLs for WhatsApp chats.

```
>>> from whatstk.data import whatsapp_urls
```

POKEMON

Brief fictional chat with Pokemon characters, which was manually designed by @lucasrodes in commit 666d6ea9cc030c4322fbe44ae64b8f1a0fdb5169.

```
>>> from whatstk.data import whatsapp_urls
>>> from whatstk import WhatsAppChat
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.POKEMON)
>>> chat.df.head(5)
                 date     username                                              message
0 2016-08-06 13:23:00  Ash Ketchum                                           Hey guys!
1 2016-08-06 13:25:00        Brock                 Hey Ash, good to have a common group!
2 2016-08-06 13:30:00        Misty  Hey guys! Long time haven't heard anything fro...
3 2016-08-06 13:45:00  Ash Ketchum  Indeed. I think having a whatsapp group nowada...
4 2016-08-06 14:30:00        Misty                                           Definetly
```

See also:

Chat file

LOREM

Chat with 500 interventions of fictional users, generated using python-lorem library.

```
>>> from whatstk.data import whatsapp_urls
>>> from whatstk import WhatsAppChat
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> chat.df.head(5)
                 date        username                                                ␣
↪message
0 2020-01-15 02:22:56            Mary                      Nostrud exercitation magna␣
↪id.
1 2020-01-15 03:33:01            Mary     Non elit irure irure pariatur exercitation.
2 2020-01-15 04:18:42  +1 123 456 789  Exercitation esse lorem reprehenderit ut ex ve.
↪..
3 2020-01-15 06:05:14        Giuseppe  Aliquip dolor reprehenderit voluptate dolore e.
↪..
4 2020-01-15 06:56:00            Mary              Ullamco duis et commodo␣
↪exercitation.
```

See also:

Chat file

LOREM1

Chat with 300 interventions of fictional users, generated using python-lorem.

```
>>> from whatstk.data import whatsapp_urls
>>> from whatstk import WhatsAppChat
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM1)
>>> chat.df.head(5)
                 date        username                                                ␣
↪message
0 2019-10-20 10:16:00            John        Laborum sed excepteur id eu cillum sunt␣
↪ut.
```

(continues on next page)

(continued from previous page)

```
1 2019-10-20 11:15:00            Mary  Ad aliquip reprehenderit proident est irure mo.
↪..
2 2019-10-20 12:16:00  +1 123 456 789  Nostrud adipiscing ex enim reprehenderit minim.
↪..
3 2019-10-20 12:57:00  +1 123 456 789  Deserunt proident laborum exercitation ex temp.
↪..
4 2019-10-20 17:28:00            John               Do ex dolor consequat tempor et␣
↪ex.
```

See also:

Chat file

LOREM2

Chat with 300 interventions of fictional users, generated using python-lorem.

Can be used along with **LOREM1** to test *`chat merging functionalities`* or *multiple-source loading*.

```
>>> from whatstk.data import whatsapp_urls
>>> from whatstk import WhatsAppChat
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM2)
>>> chat.df.head(5)
                 date        username                                                ␣
↪message
0 2020-06-20 10:16:00            John                Elit incididunt lorem sed␣
↪nostrud.
1 2020-06-20 11:15:00           Maria       Esse do irure dolor tempor ipsum fugiat.
2 2020-06-20 12:16:00  +1 123 456 789  Cillum anim non eu deserunt consectetur dolor .
↪..
3 2020-06-20 12:57:00  +1 123 456 789                 Non ipsum proident veniam est.
4 2020-06-20 17:28:00            John                    Dolore in cupidatat␣
↪proident.
```

See also:

Chat file

LOREM_2000

Chat with 2000 interventions of fictional users, generated using python-lorem.

```
>>> from whatstk.data import whatsapp_urls
>>> from whatstk import WhatsAppChat
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> chat.df.head(5)
                 date        username                                                ␣
↪message
0 2019-04-16 02:09:00  +1 123 456 789          Et labore proident laboris do labore␣
↪ex.
1 2019-04-16 03:01:00            Mary  Reprehenderit id aute consectetur aliquip nost.
↪..
2 2019-04-17 12:56:00            John  Amet magna officia ullamco pariatur ipsum cupi.
↪..
```

(continues on next page)

(continued from previous page)

```
3 2019-04-17 13:30:00            Mary  Cillum aute et cupidatat ipsum, occaecat lorem.
↪..
4 2019-04-17 15:09:00            John  Eiusmod irure laboris dolore anim, velit velit.
↪..
```

See also:

Chat file

For examples refer to *code examples* section.

5.3 Code examples

5.3.1 Basic examples

Load chat

Once you have *exported* a chat it is time to load it in python.

In this example we load the example LOREM chat, which is available online, using library class `WhatsAppChat`.

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
```

Once loaded, we can check some of the chat messages by accessing its attribute *df*, which is a pandas.DataFrame with columns *username* (name of user sending the message), *message* (message sent) and *date* index (timestamp of message).

```
>>> chat.df.head(5)
                    date        username                                          ␣
↪message
    0 2020-01-15 02:22:56            Mary                      Nostrud exercitation␣
↪magna id.
    1 2020-01-15 03:33:01            Mary     Non elit irure irure pariatur␣
↪exercitation.
    2 2020-01-15 04:18:42  +1 123 456 789  Exercitation esse lorem reprehenderit ut␣
↪ex ve...
    3 2020-01-15 06:05:14        Giuseppe  Aliquip dolor reprehenderit voluptate␣
↪dolore e...
    4 2020-01-15 06:56:00            Mary              Ullamco duis et commodo␣
↪exercitation.
```

Getting the start and end date of the chat can give us a good overview of the chat content.

```
>>> print(f"Start date: {chat.start_date}\nEnd date: {chat.end_date}")
Start date: 2020-01-15 02:22:56
End date: 2020-05-11 22:32:48
```

Also, getting a list with the chat members is simple

```
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

Load chat from multiple sources

You can also load a chat using multiple source files. You might want to use this when several files have been exported from the same chat over the years.

In the example below, we load chats LOREM1 and LOREM2.

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_sources(filepaths=[whatsapp_urls.LOREM1, whatsapp_urls.
↪LOREM2])
```

Rename usernames

In the example here, chat LOREM1 and chat LOREM2 contain slightly different usernames. In particular, in chat LOREM2, user *Mary* appears as *Maria* and *Maria2*:

```
>>> WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM1).users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
>>> WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM2).users
['+1 123 456 789', 'Giuseppe', 'John', 'Maria', 'Maria2']
>>> >>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Maria', 'Maria2', 'Mary']
```

To draw some conclusions based on user behaviour we would like to group *Mary*, *Maria* and *Maria2* under the same username. To fix this, we rename *Maria* and *Maria2* as *Mary*:

```
>>> chat = chat.rename_users({'Mary': ['Maria', 'Maria2']})
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

Load a chat with specific hformat

If `auto_header` option fails, you can still load your chat manually specifying the `hformat`. In the example below, we have that the `hformat='%d.%m.%y, %H:%M - %name:'`.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.POKEMON, hformat='%d.%m.%y,
↪ %H:%M - %name:')
>>> chat.df.head(5)
                 date     username                                              message
0 2016-08-06 13:23:00  Ash Ketchum                                           Hey guys!
1 2016-08-06 13:25:00        Brock              Hey Ash, good to have a common group!
2 2016-08-06 13:30:00        Misty  Hey guys! Long time haven't heard anything fro...
3 2016-08-06 13:45:00  Ash Ketchum  Indeed. I think having a whatsapp group nowada...
4 2016-08-06 14:30:00        Misty                                         Definetly
```

See also:

The header format

5.3.2 Visualisations

With `FigureBuilder` you can get great insights from your chat. Below we provide some examples on the visualizations that you can get with this library with the help of plotly.

Counting user interventions

Counting the user interventions can give relevant insights on which users "dominate" the conversation, even more in a group chat. To this end, object `FigureBuilder` has the method `user_interventions_count_linechart`, which generates a plotly figure with the count of user interventions.

First of all, we load a chat and create an instance of `FigureBuilder`.

```
>>> from whatstk import WhatsAppChat, FigureBuilder
>>> from whatstk.graph import plot
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> fb = FigureBuilder(chat=chat)
```

Count of user interventions

Default call of the aforementioned method displays the number of interventions sent by each user per day.

```
>>> fig = fb.user_interventions_count_linechart()
>>> plot(fig)
```

As seen in previous plot, the number of messages sent per user in a day tends to oscilate quite a lot from day to day, which might difficult a good visualisation of the data. Hence, we can use `cumulative=True` to illustrate the cumulative count of interventions instead.

```
>>> fig = fb.user_interventions_count_linechart(cumulative=True, title='User␣
↪inteventions count (cumulative)')
>>> plot(fig)
```

Additionally, we can obtain the counts for all users combined using `all_users=True`:

```
>>> fig = fb.user_interventions_count_linechart(cumulative=True, all_users=True,␣
↪title='Inteventions count (cumulative)')
>>> plot(fig)
```

Count of characters sent per user

Now, instead of counting the number of interventions we might want to explore the number of characters sent. Note that a user might send tons of messages with few words, whereas another user might send few messages with tons of words. Depending on your analysis you might prefer exploring interventions or number of characters. Getting the number of characters sent per user can be done using `msg_len=True` when calling function `user_interventions_count_linechart`.

In the following we explore the cumulative number of characters sent per user.

```
>>> fig = fb.user_interventions_count_linechart(msg_len=True, cumulative=True, title=
↪'Characters sent by user (cumulative)')
>>> plot(fig)
```

Other insights

Method `user_interventions_count_linechart` has the argument `date_mode`, which allows for several types of count-grouping methods. By default, the method obtains the counts per date (what has been used in previous examples).

Using `date_mode=hour` illustrates the distribution of user interventions over the 24 hours in a day. In this example, for instance, Giuseppe has their interventions peak in hour ranges [01:00, 02:00] and [20:00, 21:00], with 21 interventions in each.

```
>>> fig = fb.user_interventions_count_linechart(date_mode='hour', title='User␣
↪interventions count (hour)',
xlabel='Hour')
>>> plot(fig)
```

Using `date_mode=weekday` illustrates the distribution of user interventions over the 7 days of the week. In this example, for instance, we see that Monday and Sunday are the days with the most interventions.

```
>>> fig = fb.user_interventions_count_linechart(date_mode='weekday', title='User␣
↪interventions count (weekly)',
xlabel='Week day')
>>> plot(fig)
```

Using `date_mode=month` illustrates the distribution of user interventions over the 12 months of the year. In this example, for instance, we observe that all users have their interventions peak in June (except for Giuseppe, which has their peak in July). Maybe summer calling?

```
>>> fig = fb.user_interventions_count_linechart(date_mode='month', title='User␣
↪interventions count (yearly)', xlabel='Month')
>>> plot(fig)
```

Message length boxplot

Different users send different sort of messages. In particular, the length of the messages (number of characters) can substatially vary depending on the user sending the message.

In this example, we explore the statistics behind the length of user messages. To this end, we can use method `user_msg_length_boxplot`, which illustrates the length of each user's messages by means of box plots.

```
>>> from whatstk import WhatsAppChat, FigureBuilder
>>> from whatstk.graph import plot
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> fig = FigureBuilder(chat=chat).user_msg_length_boxplot()
>>> plot(fig)
```

User interaction

The user interaction can shed some light on the different kinds of conversations that occur in a chat group. For instance, when a certain topic appears some users might intervene and others will not, forming *user clusters*. To this end, a first approach in detecting such clusters resides in which users respond to which users.

User interaction heatmap

In the following we visualize the *response matrix*, which tells us the number of messages sent by a certain user to the rest of users.

For instance, in this specific example we observe that user *Giuseppe* sends 153 messages to *+ 1 123 456 789* and that *Mary* receives 122 messages from *John*.

```
>>> from whatstk import WhatsAppChat, FigureBuilder
>>> from whatstk.graph import plot
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> fig = FigureBuilder(chat=chat).user_message_responses_heatmap()
>>> plot(fig)
```

See also:

- `user_message_responses_heatmap`

User interaction flow

A good way o visualize responses between users are Sankey diagrams. The information conveyed by the graph below is the same as the one in previous section, but the way it is done is slightly different (sankey diagram instead of a heatmap).

```
>>> from whatstk import WhatsAppChat, FigureBuilder
>>> from whatstk.graph import plot
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> fig = FigureBuilder(chat=chat).user_message_responses_flow()
>>> plot(fig)
```

See also:

- `user_message_responses_flow`

Custom plot

`FigureBuilder` provides some tools to easily visualize your chat. However, the possible visualizations are infinite. Here, we provide some examples of a custom visualization using some library tools together with pandas and plotly.

Number of messages vs. Number of characters sent

For each user, we will obtain a 2D scatter plot measuring the number of messages and characters sent in a day. That is, for a given user we will have N points, where N is the number of days that the user has sent at least one message. Each point therefore corresponds to a specific day, where the x-axis and the y-axis measure the number of messages sent and the average number of characters per message in that day, respectively.

First of all, lets instatiate objects `WhatsAppChat` (chat loading) and `FigureBuilder` (figure coloring).

```
>>> from whatstk import WhatsAppChat, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM_2000)
>>> fb = FigureBuilder(chat=chat)
```

Next, we obtain the number of messages and number of characters sent per user per day.

```
>>> from whatstk.analysis import get_interventions_count
>>> counts_interv = get_interventions_count(chat=chat, date_mode='date', msg_
↪length=False, cumulative=False)
>>> counts_len = get_interventions_count(chat=chat, date_mode='date', msg_length=True,
↪ cumulative=False)
```

Time to process a bit the data. We obtain a DataFrame with five columns: *username*, *date*, *num_characters*, *num_interventions* and *avg_characters*.

```
>>> import pandas as pd
>>> counts_len = pd.DataFrame(counts_len.unstack(), columns=['num_characters'])
>>> counts_interv = pd.DataFrame(counts_interv.unstack(), columns=['num_interventions
↪'])
>>> counts = counts_len.merge(counts_interv, left_index=True, right_index=True)
>>> # Remove all zero entries and get average number of characters
>>> counts = counts[~(counts['num_interventions'] == 0)].reset_index()
>>> counts['avg_characters'] = counts['num_characters']/counts['num_interventions']
>>> counts.head(5)
         username       date  num_characters  num_interventions  avg_characters
0  +1 123 456 789 2019-04-16              40                  1       40.000000
1  +1 123 456 789 2019-04-17              21                  1       21.000000
2  +1 123 456 789 2019-04-21              90                  2       45.000000
3  +1 123 456 789 2019-04-25             127                  3       42.333333
4  +1 123 456 789 2019-04-26              33                  1       33.000000

[5 rows x 5 columns]
```

So far we have obtained a dataframe `counts`, whose rows correspond to a specific message. However, in this example we are interested in the aggregated values per day. Hence, we group this dataframe by user and date and re-calculate the number of messages sent and average number of characters sent per day.

```
>>> agg_operations = {'avg_characters': 'mean','num_interventions': 'mean'}
>>> counts = counts.groupby(['username', counts.date.dt.date]).agg(agg_operations)
>>> counts = counts.rename_axis(index=['username', 'date'])
>>> counts = counts.reset_index()
>>> counts.head(5)
          username        date      avg_characters  num_interventions
0   +1 123 456 789  2019-04-16           40.000000                  1
1   +1 123 456 789  2019-04-17           21.000000                  1
2   +1 123 456 789  2019-04-21           45.000000                  2
3   +1 123 456 789  2019-04-25           42.333333                  3
4   +1 123 456 789  2019-04-26           33.000000                  1
```

Once the dataframe is obtained, we generate a plot using Histogram2dContour by plotly.

```
>>> from whatstk.graph import plot
>>> import plotly.graph_objs as go
>>> traces = []
>>> for username in fb.usernames:
>>>     counts_user = counts[counts['username']==username]
>>>     traces.append(
>>>         go.Histogram2dContour(
>>>             contours={'coloring': 'none'},
>>>             x=counts_user['num_interventions'],
>>>             y=counts_user['avg_characters'],
>>>             # mode='markers',
>>>             # marker=dict(color=fb.user_color_mapping[username], opacity=0.2),
>>>             name=username,
>>>             showlegend=True,
>>>             line={'color': fb.user_color_mapping[username]},
>>>             nbinsx=10, nbinsy=20
>>>         )
>>>     )
```

```
>>> layout = {
>>>     'title': 'Average number of characters sent in a day vs Interventions per day
↪',
>>>     'yaxis_title': 'avg characters',
>>>     'xaxis_title': 'num interventions',
>>> }
>>> fig = go.Figure(data=traces, layout=layout)
>>> plot(fig)
```

If you think that something is missing please raise an issue.

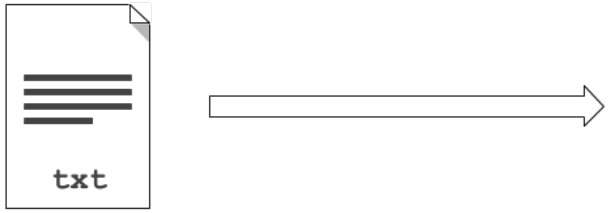
5.4 API Reference

5.4.1 Main objects

WhatsAppChat

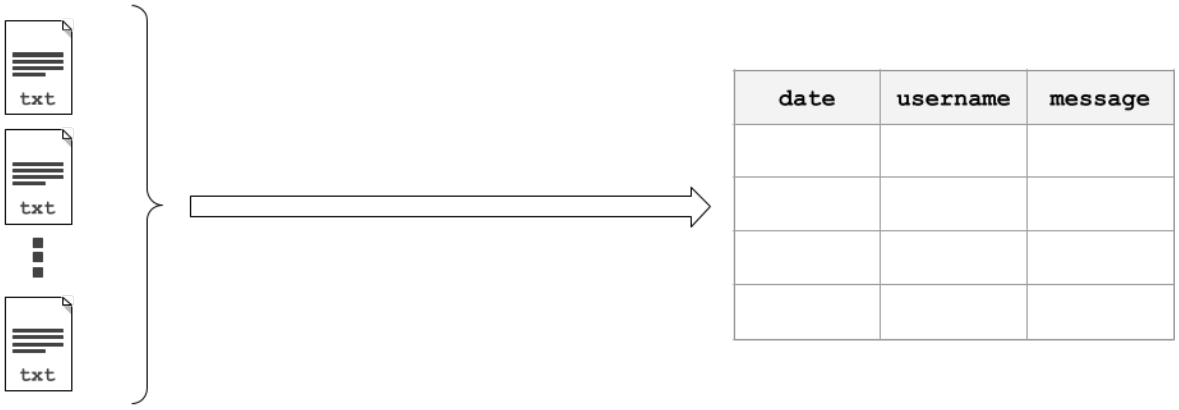
Object `WhatsAppChat` works as a bridge between the python code and the whatsapp chat text file. Easily load a chat from a text file and work with it using all the power of pandas.

A chat can be loaded from a single source file using `WhatsAppChat.from_source`



date	username	message

or multiple source files using `WhatsAppChat.from_sources`



class whatstk.**WhatsAppChat**(*df*)

Bases: *whatstk._chat.BaseChat*

Load and process a WhatsApp chat file.

Parameters **df** (*pandas.DataFrame*) – Chat.

Attributes

df	Chat as DataFrame.
end_date	Chat end date.
start_date	Chat starting date.
users	List with users.

Methods

from_source(filepath, **kwargs)	Create an instance from a chat text file.
from_sources(filepaths[, auto_header, ...])	Load a WhatsAppChat instance from multiple sources.
merge(chat[, rename_users])	Merge current instance with `chat`.
rename_users(mapping)	Rename users.
to_csv(filepath)	Save chat as csv.
to_txt(filepath[, hformat])	Export chat to a text file.

Example

This simple example loads a chat using `WhatsAppChat`. Once loaded, we can access its attribute *df*, which contains the loaded chat as a DataFrame.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.POKEMON)
>>> chat.df.head(5)
                 date     username
↪message
```

(continues on next page)

(continued from previous page)

```
0 2016-08-06 13:23:00  Ash Ketchum                                          Hey␣
↪guys!
1 2016-08-06 13:25:00        Brock              Hey Ash, good to have a common␣
↪group!
2 2016-08-06 13:30:00        Misty  Hey guys! Long time haven't heard anything␣
↪fro...
3 2016-08-06 13:45:00  Ash Ketchum  Indeed. I think having a whatsapp group␣
↪nowada...
4 2016-08-06 14:30:00        Misty                                           ␣
↪Definetly
```

property df
Chat as DataFrame.

Returns pandas.DataFrame

property end_date
Chat end date.

Returns datetime

classmethod from_source (*filepath*, ***kwargs*)
Create an instance from a chat text file.

Parameters

- **filepath** (*str*) – Path to the file. It can be a local file (e.g. 'path/to/file.txt') or an URL to a hosted file (e.g. 'http://www.url.to/file.txt')
- ****kwargs** – Refer to the docs from `df_from_txt_whatsapp` for details on additional arguments.

Returns *WhatsAppChat* – Class instance with loaded and parsed chat.

See also:

- `df_from_txt_whatsapp`
- `WhatsAppChat.from_sources`

classmethod from_sources (*filepaths*, *auto_header=None*, *hformat=None*, *encoding='utf-8'*)
Load a WhatsAppChat instance from multiple sources.

Parameters

- **filepaths** (*list*) – List with filepaths.
- **auto_header** (*bool, optional*) – Detect header automatically (applies to all files). If None, attempts to perform automatic header detection for all files. If False, `hformat` is required.
- **hformat** (*list, optional*) – List with the *header format* to be used for each file. The list must be of length equal to `len(filenames)`. A valid header format might be '[%y-%m-%d %H:%M:%S] - %name:'.
- **encoding** (*str*) – Encoding to use for UTF when reading/writing (ex. 'utf-8'). List of Python standard encodings.

Returns *WhatsAppChat* – Class instance with loaded and parsed chat.

See also:

- `WhatsAppChat.from_source`
- `merge_chats`

Example

Load a chat using two text files. In this example, we use sample chats (available online, see urls in source code `whatstk.data`).

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.LOREM1
>>> filepath_2 = whatsapp_urls.LOREM2
>>> chat = WhatsAppChat.from_sources(filepaths=[filepath_1, filepath_2])
>>> chat.df.head(5)
                 date        username                                                ␣
↪   message
0 2019-10-20 10:16:00            John        Laborum sed excepteur id eu␣
↪cillum sunt ut.
1 2019-10-20 11:15:00            Mary  Ad aliquip reprehenderit proident est␣
↪irure mo...
2 2019-10-20 12:16:00  +1 123 456 789  Nostrud adipiscing ex enim␣
↪reprehenderit minim...
3 2019-10-20 12:57:00  +1 123 456 789  Deserunt proident laborum exercitation␣
↪ex temp...
4 2019-10-20 17:28:00            John               Do ex dolor consequat␣
↪tempor et ex.
```

merge(*chat*, *rename_users=None*)

Merge current instance with `chat`.

Parameters

- **chat** (*WhatsAppChat*) – Another chat.
- **rename_users** (*dict*) – Dictionary mapping old names to new names. Example: {'John':['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *WhatsAppChat* – Merged chat.

See also:

- *rename_users*
- `merge_chats`

Example

Merging two chats can become handy when you have exported a chat in different times with your phone and hence each exported file might contain data that is unique to that file.

In this example however, we merge files from different chats.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.LOREM1
>>> filepath_2 = whatsapp_urls.LOREM2
```

(continues on next page)

(continued from previous page)

```
>>> chat_1 = WhatsAppChat.from_source(filepath=filepath_1)
>>> chat_2 = WhatsAppChat.from_source(filepath=filepath_2)
>>> chat = chat_1.merge(chat_2)
```

rename_users (*mapping*)
Rename users.

This might be needed in multiple occations:

- Change typos in user names stored in phone.
- **If a user appears multiple times with different usernames, group these under the same name (this might** happen when multiple chats are merged).

Parameters **mapping** (*dict*) – Dictionary mapping old names to new names, example: {'John': ['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *pandas.DataFrame* – DataFrame with users renamed according to *mapping*.

Raises **ValueError** – Raised if mapping is not correct.

Examples

Load LOREM2 chat and rename users *Maria* and *Maria2* to *Mary*.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM2)
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Maria', 'Maria2']
>>> chat = chat.rename_users(mapping={'Mary': ['Maria', 'Maria2']})
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

property start_date
Chat starting date.

Returns datetime

to_csv (*filepath*)
Save chat as csv.

Parameters **filepath** (*str*) – Name of file.

to_txt (*filepath*, *hformat=None*)
Export chat to a text file.

Usefull to export the chat to different formats (i.e. using different hformats).

Parameters

- **filepath** (*str*) – Name of the file to export (must be a local path).
- **hformat** (*str, optional*) – Header format. Defaults to '%y-%m-%d, %H:%M - %name:'.

property users
List with users.

Returns list

FigureBuilder

whatstk provides this object to ease the generation of insightfull plots from your chat. `FigureBuilder` contains several methods to generate different plots. It assigns a unique color to each user, so that a user can be easily identified in all plots.

To insantiate it, you just need to provide the chat (as pandas.DataFrame or *BaseChat*-API-compliant object).

class whatstk.**FigureBuilder**(*df=None*, *chat=None*)

Bases: `object`

Generate a variety of figures from your loaded chat.

Integrates feature extraction and visualization logic to automate data plots.

Note: Either `df` or `chat` must be provided.

Parameters

- **df** (*pandas.DataFrame, optional*) – Chat data. Atribute *df* of a chat loaded using Chat. If a value is given, `chat` is ignored.
- **chat** (*Chat, optional*) – Chat data. Object obtained when chat loaded using Chat. Required if `df` is None.

Attributes

`user_color_mapping`	Get mapping between user and color.
`usernames`	Get list with users available in given chat.

Methods

`user_interventions_count_linechart`([…])	Plot number of user interventions over time.
`user_message_responses_flow`([title])	Get the flow of message responses.
`user_message_responses_heatmap`([norm, title])	Get the response matrix heatmap.
`user_msg_length_boxplot`([title, xlabel])	Generate figure with boxplots of each user's message length.

property user_color_mapping

Get mapping between user and color.

Each user is assigned a color automatically, so that this color is preserved for that user in all to-be-generated plots.

Returns *dict* – Mapping from username to color (rgb).

user_interventions_count_linechart(*date_mode='date'*, *msg_length=False*, *cumulative=False*, *all_users=False*, *title='User interventions count'*, *xlabel='Date/Time'*, *cummulative=None*)

Plot number of user interventions over time.

Parameters

- **date_mode** (*str, optional*) – Choose mode to group interventions by. Defaults to `'date'`. Available modes are:
  - `'date'`: Grouped by particular date (year, month and day).
  - `'hour'`: Grouped by hours.

- `'month'`: Grouped by months.
- `'weekday'`: Grouped by weekday (i.e. monday, tuesday, . . . , sunday).
- `'hourweekday'`: Grouped by weekday and hour.

- **msg_length** (*bool, optional*) – Set to True to count the number of characters instead of number of messages sent.
- **cumulative** (*bool, optional*) – Set to True to obtain commulative counts.
- **all_users** (*bool, optional*) – Obtain number of interventions of all users combined. Defaults to False.
- **title** (*str, optional*) – Title for plot. Defaults to "User interventions count".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to "Date/Time".
- **cummulative** (*bool, optional*) – Deprecated, use cumulative.

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_interventions_count`
- `fig_scatter_time`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_interventions_count_
↪linechart(cumulative=True)
>>> plot(fig)
```

user_message_responses_flow(*title='Message flow'*)

Get the flow of message responses.

A response from user X to user Y happens if user X sends a message right after a message from user Y.

Uses a Sankey diagram.

Parameters **title** (*str, optional*) – Title for plot. Defaults to "Message flow".

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_response_matrix`
- `fig_sankey`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_message_responses_flow()
>>> plot(fig)
```

user_message_responses_heatmap(*norm='absolute'*, *title='Response matrix'*)

Get the response matrix heatmap.

A response from user X to user Y happens if user X sends a message right after a message from user Y.

Parameters

- **norm** (*str, optional*) – Specifies the type of normalization used for reponse count. Can be:
  - `'absolute'`: Absolute count of messages.
  - `'joint'`: Normalized by total number of messages sent by all users.
  - `'sender'`: Normalized per sender by total number of messages sent by user.
  - `'receiver'`: Normalized per receiver by total number of messages sent by user.
- **title** (*str, optional*) – Title for plot. Defaults to "Response matrix".

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_response_matrix`
- `fig_heatmap`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_message_responses_heatmap()
>>> plot(fig)
```

user_msg_length_boxplot(*title='User message length'*, *xlabel='User'*)

Generate figure with boxplots of each user's message length.

Parameters

- **title** (*str, optional*) – Title for plot. Defaults to "User message length".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to "User".

Returns *dict* – Dictionary with data and layout. Plotly compatible.

See also:

- `fig_boxplot_msglen`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_msg_length_boxplot()
>>> plot(fig)
```

property usernames
Get list with users available in given chat.

Returns *list* – List with usernames available in chat DataFrame.

5.4.2 Core API

whatstk.whatsapp

WhatsApp parser.

---

whatstk.whatsapp.objects

Library WhatsApp objects.

Classes

WhatsAppChat(df)	Load and process a WhatsApp chat file.

class whatstk.whatsapp.objects.**WhatsAppChat**(*df*)
Bases: *whatstk._chat.BaseChat*

Load and process a WhatsApp chat file.

Parameters **df** (*pandas.DataFrame*) – Chat.

Attributes

df	Chat as DataFrame.
end_date	Chat end date.
start_date	Chat starting date.
users	List with users.

Methods

from_source(filepath, **kwargs)	Create an instance from a chat text file.
from_sources(filepaths[, auto_header, ...])	Load a WhatsAppChat instance from multiple sources.
merge(chat[, rename_users])	Merge current instance with chat.
rename_users(mapping)	Rename users.
to_csv(filepath)	Save chat as csv.

continues on next page

Table 8 – continued from previous page

`to_txt`(filepath[, hformat])	Export chat to a text file.

Example

This simple example loads a chat using `WhatsAppChat`. Once loaded, we can access its attribute *df*, which contains the loaded chat as a DataFrame.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.POKEMON)
>>> chat.df.head(5)
                 date     username                                                 ␣
↪message
0 2016-08-06 13:23:00  Ash Ketchum                                                Hey␣
↪guys!
1 2016-08-06 13:25:00        Brock              Hey Ash, good to have a common␣
↪group!
2 2016-08-06 13:30:00        Misty  Hey guys! Long time haven't heard anything␣
↪fro...
3 2016-08-06 13:45:00  Ash Ketchum  Indeed. I think having a whatsapp group␣
↪nowada...
4 2016-08-06 14:30:00        Misty                                                 ␣
↪Definetly
```

property df
Chat as DataFrame.

Returns pandas.DataFrame

property end_date
Chat end date.

Returns datetime

classmethod from_source (*filepath*, ***kwargs*)
Create an instance from a chat text file.

Parameters

- **filepath** (*str*) – Path to the file. It can be a local file (e.g. 'path/to/file.txt') or an URL to a hosted file (e.g. 'http://www.url.to/file.txt')
- ****kwargs** – Refer to the docs from `df_from_txt_whatsapp` for details on additional arguments.

Returns *WhatsAppChat* – Class instance with loaded and parsed chat.

See also:

- `df_from_txt_whatsapp`
- `WhatsAppChat.from_sources`

classmethod from_sources (*filepaths*, *auto_header=None*, *hformat=None*, *encoding='utf-8'*)
Load a WhatsAppChat instance from multiple sources.

Parameters

- **filepaths** (*list*) – List with filepaths.

- **auto_header** (*bool, optional*) – Detect header automatically (applies to all files). If None, attempts to perform automatic header detection for all files. If False, `hformat` is required.
- **hformat** (*list, optional*) – List with the *header format* to be used for each file. The list must be of length equal to `len(filenames)`. A valid header format might be '[%y-%m-%d %H:%M:%S] - %name:'.
- **encoding** (*str*) – Encoding to use for UTF when reading/writing (ex. 'utf-8'). List of Python standard encodings.

Returns *WhatsAppChat* – Class instance with loaded and parsed chat.

See also:

- `WhatsAppChat.from_source`
- `merge_chats`

Example

Load a chat using two text files. In this example, we use sample chats (available online, see urls in source code `whatstk.data`).

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.LOREM1
>>> filepath_2 = whatsapp_urls.LOREM2
>>> chat = WhatsAppChat.from_sources(filepaths=[filepath_1, filepath_2])
>>> chat.df.head(5)
                 date        username                                                ␣
↪   message
0 2019-10-20 10:16:00            John        Laborum sed excepteur id eu␣
↪cillum sunt ut.
1 2019-10-20 11:15:00            Mary  Ad aliquip reprehenderit proident est␣
↪irure mo...
2 2019-10-20 12:16:00  +1 123 456 789  Nostrud adipiscing ex enim␣
↪reprehenderit minim...
3 2019-10-20 12:57:00  +1 123 456 789  Deserunt proident laborum exercitation␣
↪ex temp...
4 2019-10-20 17:28:00            John               Do ex dolor consequat␣
↪tempor et ex.
```

merge(*chat*, *rename_users=None*)

Merge current instance with `chat`.

Parameters

- **chat** (*WhatsAppChat*) – Another chat.
- **rename_users** (*dict*) – Dictionary mapping old names to new names. Example: {'John':['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *WhatsAppChat* – Merged chat.

See also:

- *rename_users*
- `merge_chats`

Example

Merging two chats can become handy when you have exported a chat in different times with your phone and hence each exported file might contain data that is unique to that file.

In this example however, we merge files from different chats.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.LOREM1
>>> filepath_2 = whatsapp_urls.LOREM2
>>> chat_1 = WhatsAppChat.from_source(filepath=filepath_1)
>>> chat_2 = WhatsAppChat.from_source(filepath=filepath_2)
>>> chat = chat_1.merge(chat_2)
```

rename_users (*mapping*)
Rename users.

This might be needed in multiple occations:

- Change typos in user names stored in phone.
- **If a user appears multiple times with different usernames, group these under the same name (this might** happen when multiple chats are merged).

Parameters **mapping** (*dict*) – Dictionary mapping old names to new names, example: {'John': ['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *pandas.DataFrame* – DataFrame with users renamed according to *mapping*.

Raises **ValueError** – Raised if mapping is not correct.

Examples

Load LOREM2 chat and rename users *Maria* and *Maria2* to *Mary*.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM2)
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Maria', 'Maria2']
>>> chat = chat.rename_users(mapping={'Mary': ['Maria', 'Maria2']})
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

property start_date
Chat starting date.

Returns datetime

to_csv (*filepath*)
Save chat as csv.

Parameters **filepath** (*str*) – Name of file.

to_txt (*filepath*, *hformat=None*)
Export chat to a text file.

Usefull to export the chat to different formats (i.e. using different hformats).

Parameters

- **filepath** (*str*) – Name of the file to export (must be a local path).
- **hformat** (*str, optional*) – Header format. Defaults to '%y-%m-%d, %H:%M - %name:'.

property users
List with users.

Returns list

---

whatstk.whatsapp.parser

Parser utils.

Functions

df_from_txt_whatsapp(filepath[, ...])	Load chat as a DataFrame.
generate_regex(hformat)	Generate regular expression from hformat.

whatstk.whatsapp.parser.**df_from_txt_whatsapp**(*filepath*, *auto_header=True*, *hformat=None*, *encoding='utf-8'*)

Load chat as a DataFrame.

Parameters

- **filepath** (*str*) – Path to the file. It can be a local file (e.g. 'path/to/file.txt') or an URL to a hosted file (e.g. 'http://www.url.to/file.txt')
- **auto_header** (*bool, optional*) – Detect header automatically. If False, `hformat` is required.
- **hformat** (*str, optional*) – *Format of the header*, e.g. `'[%y-%m-%d %H:%M:%S] - %name:'`. Use following keywords:
  - `'%y'`: for year (`'%Y'` is equivalent).
  - `'%m'`: for month.
  - `'%d'`: for day.
  - `'%H'`: for 24h-hour.
  - `'%I'`: for 12h-hour.
  - `'%M'`: for minutes.
  - `'%S'`: for seconds.
  - `'%P'`: for "PM"/"AM" or "p.m."/"a.m." characters.
  - `'%name'`: for the username.

  Example 1: For the header '12/08/2016, 16:20 - username:' we have the `'hformat='%d/%m/%y, %H:%M - %name:'`.

  Example 2: For the header '2016-08-12, 4:20 PM - username:' we have `hformat='%y-%m-%d, %I:%M %P - %name:'`.

- **encoding** (*str, optional*) – Encoding to use for UTF when reading/writing (ex. 'utf-8'). List of Python standard encodings.

Returns *WhatsAppChat* – Class instance with loaded and parsed chat.

See also:

- `WhatsAppChat.from_source`
- `extract_header_from_text`

whatstk.whatsapp.parser.**generate_regex**(*hformat*)

Generate regular expression from hformat.

Parameters **hformat** (*str*) – Simplified syntax for the header, e.g. `'%y-%m-%d, %H:%M:%S - %name:'`.

Returns *str* – Regular expression corresponding to the specified syntax.

Example

Generate regular expression corresponding to `'hformat=%y-%m-%d, %H:%M:%S - %name:'`.

```
>>> from whatstk.whatsapp.parser import generate_regex
>>> generate_regex('%y-%m-%d, %H:%M:%S - %name:')
('(?P<year>\\d{2,4})-(?P<month>\\d{1,2})-(?P<day>\\d{1,2}), (?P<hour>\\d{1,2}):(?P
↪<minutes>\\d{2}):(?
P<seconds>\\d{2}) - (?P<username>[^:]*): ', '(?P<year>\\d{2,4})-(?P<month>\\d{1,2}
↪)-(?P<day>\\d{1,2}), (?
P<hour>\\d{1,2}):(?P<minutes>\\d{2}):(?P<seconds>\\d{2}) - ')
```

whatstk.whatsapp.auto_header

Detect header from chat.

Functions

`extract_header_from_text`(text[, encoding])	Extract header from text.

whatstk.whatsapp.auto_header.**extract_header_from_text**(*text*, *encoding='utf-8'*)

Extract header from text.

Parameters

- **text** (*str*) – Loaded chat as string (whole text).
- **encoding** (*str*) – Encoding to use for UTF when reading/writing (ex. 'utf-8'). List of Python standard encodings.

Returns *str* – Format extracted. None if no header was extracted.

Example

Load a chat using two text files. In this example, we use sample chats (available online, see urls in source code `whatstk.data`).

```
>>> from whatstk.whatsapp.parser import extract_header_from_text
>>> from urllib.request import urlopen
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.POKEMON
>>> with urlopen(filepath_1) as f:
...     text = f.read().decode('utf-8')
>>> extract_header_from_text(text)
'%d.%m.%y, %H:%M - %name:
```

whatstk.whatsapp.generation

Automatic generation of chat using Lorem Ipsum text and time series statistics.

Classes

`ChatGenerator`(size[, users, seed])	Generate a chat.

Functions

`generate_chats_hformats`(output_path[, size, ...])	Generate a chat and export using given header format.

class `whatstk.whatsapp.generation.`**ChatGenerator**(*size*, *users=None*, *seed=100*)

Bases: `object`

Generate a chat.

Parameters

- **size** (*int*) – Number of messages to generate.
- **users** (*list, optional*) – List with names of the users. Defaults to module variable USERS.
- **seed** (*int, optional*) – Seed for random processes. Defaults to 100.

Methods

`generate`([filepath, hformat, last_timestamp])	Generate random chat as `WhatsAppChat`.

Examples

This simple example loads a chat using `WhatsAppChat`. Once loaded, we can access its attribute `df`, which contains the loaded chat as a DataFrame.

```
>>> from whatstk.whatsapp.generation import ChatGenerator
>>> from datetime import datetime
>>> from whatstk.data import whatsapp_urls
>>> chat = ChatGenerator(size=10).generate(last_timestamp=datetime(2020, 1, 1, 0,␣
↪0))
>>> chat.df.head(5)
                        date  username                                            ␣
↪message
0 2019-12-31 09:43:04.000525  Giuseppe                                 Nisi ad esse␣
↪cillum.
1 2019-12-31 10:19:21.980039  Giuseppe      Tempor dolore sint in eu lorem veniam␣
↪veniam.
2 2019-12-31 13:56:45.575426  Giuseppe  Do quis fugiat sint ut ut, do anim eu est␣
↪qui ...
3 2019-12-31 15:47:29.995420  Giuseppe  Do qui qui elit ea in sed culpa, aliqua␣
↪magna ...
4 2019-12-31 16:23:00.348542      Mary  Sunt excepteur mollit voluptate dolor␣
↪sint occ...
```

generate (*filepath=None*, *hformat=None*, *last_timestamp=None*)

Generate random chat as `WhatsAppChat`.

Parameters

- **filepath** (*str*) – If given, generated chat is saved with name `filepath` (must be a local path).
- **hformat** (*str, optional*) – *Format of the header*, e.g. `'[%y-%m-%d %H:%M:%S] - %name:'`.
- **last_timestamp** (*datetime, optional*) – Datetime of last message. If *None*, defaults to current date.

Returns *WhatsAppChat* – Chat with random messages.

See also:

- `WhatsAppChat.to_txt`

whatstk.whatsapp.generation.**generate_chats_hformats**(*output_path*, *size=2000*, *hformats=None*, *filepaths=None*, *last_timestamp=None*, *seed=100*, *verbose=False*)

Generate a chat and export using given header format.

If no hformat specified, chat is generated & exported using all supported header formats.

Parameters

- **output_path** (*str*) – Path to directory to export all generated chats as txt.
- **size** (*int, optional*) – Number of messages of the chat. Defaults to 2000.
- **hformats** (*list, optional*) – List of header formats to use when exporting chat. If None, defaults to all supported header formats.

- **filepaths** (*list, optional*) – List with filepaths. If None, defaults to *hformat.replace(' ', '_').replace('/', '')*.
- **last_timestamp** (*datetime, optional*) – Datetime of last message. If *None*, defaults to current date.
- **seed** (*int, optional*) – Seed for random processes. Defaults to 100.
- **verbose** (*bool*) – Set to True to print runtime messages.

See also:

- `ChatGenerator`
- `ChatGenerator.generate`

---

whatstk.whatsapp.hformat

Header format utils.

Example: Check if header is available.

```
>>> from whatstk.utils.hformat import is_supported
>>> is_supported('%y-%m-%d, %H:%M:%S - %name:')
(True, True)
```

Functions

`get_supported_hformats_as_dict`()	Get dictionary with supported formats and relevant info.
`get_supported_hformats_as_list`()	Get list of supported formats.
`is_supported`(hformat)	Check if header *hformat* is currently supported.
`is_supported_verbose`(hformat)	Check if header *hformat* is currently supported (both manually and using *auto_header*).

`whatstk.whatsapp.hformat.`**`get_supported_hformats_as_dict`**`()`
Get dictionary with supported formats and relevant info.

Returns

dict –

Dict with two elements:

- `format`: Header format. All formats appearing are supported.
- `auto_header`: 1 if auto_header is supported), 0 otherwise.

`whatstk.whatsapp.hformat.`**`get_supported_hformats_as_list`**`()`
Get list of supported formats.

Returns *list* – List with supported formats (as str).

`whatstk.whatsapp.hformat.`**`is_supported`**`(`*hformat*`)`
Check if header *hformat* is currently supported.

Parameters **hformat** (*str*) – Header format.

Returns *tuple* – * bool: True if header is supported. * bool: True if header is supported with *auto_header* feature.

whatstk.whatsapp.hformat.**is_supported_verbose**(*hformat*)

Check if header *hformat* is currently supported (both manually and using *auto_header*).

Result is shown as a string.

Parameters **hformat** (*str*) – Information message.

Example

Check if format `'%y-%m-%d, %H:%M - %name:'` is supported.

```
>>> from whatstk.whatsapp.hformat import is_supported_verbose
>>> is_supported_verbose('%y-%m-%d, %H:%M - %name:')
"The header '%y-%m-%d, %H:%M - %name:' is supported. `auto_header` for this␣
↪header is supported."
```

whatstk.analysis

Analysis tools.

Functions

get_interventions_count([df, chat, ...])	Get number of interventions per user per unit of time.
get_response_matrix([df, chat, zero_own, norm])	Get response matrix for given chat.

whatstk.analysis.**get_interventions_count**(*df=None*, *chat=None*, *date_mode='date'*, *msg_length=False*, *cumulative=False*, *all_users=False*, *cummulative=None*)

Get number of interventions per user per unit of time.

The unit of time can be chosen by means of argument `date_mode`.

Note: Either `df` or `chat` must be provided.

Parameters

- **df** (*pandas.DataFrame, optional*) – Chat data. Atribute *df* of a chat loaded using Chat. If a value is given, `chat` is ignored.
- **chat** (*Chat, optional*) – Chat data. Object obtained when chat loaded using Chat. Required if `df` is None.
- **date_mode** (*str, optional*) – Choose mode to group interventions by. Defaults to `date_mode=date`. Available modes are:
  - `'date'`: Grouped by particular date (year, month and day).
  - `'hour'`: Grouped by day hours (24 hours).
  - `'month'`: Grouped by months (12 months).
  - `'weekday'`: Grouped by weekday (i.e. monday, tuesday, ..., sunday).
  - `'hourweekday'`: Grouped by weekday and hour.

- **msg_length** (*bool, optional*) – Set to True to count the number of characters instead of number of messages sent.
- **cumulative** (*bool, optional*) – Set to True to obtain commulative counts.
- **all_users** (*bool, optional*) – Obtain number of interventions of all users combined. Defaults to False.
- **cummulative** (*bool, optional*) – Deprecated, use cumulative.

Returns *pandas.DataFrame* – DataFrame with shape *NxU*, where *N*: number of time-slots and *U*: number of users.

Raises **ValueError** – if `date_mode` value is not supported.

Example

Get number of interventions per user from POKEMON chat. The counts are represented as a *NxU* matrix, where *N*: number of time-slots and *U*: number of users.

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.analysis import get_interventions_count
>>> from whatstk.data import whatsapp_urls
>>> filepath = whatsapp_urls.POKEMON
>>> chat = WhatsAppChat.from_source(filepath)
>>> counts = get_interventions_count(chat=chat, date_mode='date', msg_
↪length=False)
>>> counts.head(5)
username    Ash Ketchum  Brock  Jessie & James  ...  Prof. Oak  Raichu  Wobbuffet
date                                            ...
2016-08-06            2      2               0  ...          0       0          0
2016-08-07            1      1               0  ...          1       0          0
2016-08-10            1      0               1  ...          0       2          0
2016-08-11            0      0               0  ...          0       0          0
2016-09-11            0      0               0  ...          0       0          0

[5 rows x 8 columns]
```

whatstk.analysis.**get_response_matrix**(*df=None*, *chat=None*, *zero_own=True*, *norm='absolute'*)

Get response matrix for given chat.

Obtains a DataFrame of shape *[n_users, n_users]* counting the number of responses between members. Responses can be counted in different ways, e.g. using absolute values or normalised values. Responses are counted based solely on consecutive messages. That is, if $user_i$ sends a message right after $user_j$, it will be counted as a response from $user_i$ to $user_j$.

Axis 0 lists senders and axis 1 lists receivers. That is, the value in cell (i, j) denotes the number of times $user_i$ responded to a message from $user_j$.

Note: Either `df` or `chat` must be provided.

Parameters

- **df** (*pandas.DataFrame, optional*) – Chat data. Atribute *df* of a chat loaded using Chat. If a value is given, `chat` is ignored.
- **chat** (*Chat, optional*) – Chat data. Object obtained when chat loaded using Chat. Required if `df` is None.

- **zero_own** (*bool, optional*) – Set to True to avoid counting own responses. Defaults to True.
- **norm** (*str, optional*) – Specifies the type of normalization used for reponse count. Can be:
  - `'absolute'`: Absolute count of messages.
  - `'joint'`: Normalized by total number of messages sent by all users.
  - `'sender'`: Normalized per sender by total number of messages sent by user.
  - `'receiver'`: Normalized per receiver by total number of messages sent by user.

Returns *pandas.DataFrame* – Response matrix.

Example

Get absolute count on responses (consecutive messages) between users.

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.analysis import get_response_matrix
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.POKEMON)
>>> responses = get_response_matrix(chat=chat)
>>> responses
                Ash Ketchum  Brock  ...  Raichu  Wobbuffet
Ash Ketchum               0      0  ...       1          0
Brock                     1      0  ...       0          0
Jessie & James            0      1  ...       0          0
Meowth                    0      0  ...       0          0
Misty                     2      1  ...       1          0
Prof. Oak                 0      1  ...       0          0
Raichu                    1      0  ...       0          0
Wobbuffet                 0      0  ...       0          0
```

whatstk.graph

Plot tools using plotly.

Import plot (by plotly) to plot figures.

```
>>> from whatstk.graph import plot
```

whatstk.graph.base

Build plotly-compatible figures.

Classes

`FigureBuilder`([df, chat])	Generate a variety of figures from your loaded chat.

class whatstk.graph.base.**FigureBuilder**(*df=None*, *chat=None*)

Bases: `object`

Generate a variety of figures from your loaded chat.

Integrates feature extraction and visualization logic to automate data plots.

Note: Either `df` or `chat` must be provided.

Parameters

- **df** (*pandas.DataFrame, optional*) – Chat data. Atribute *df* of a chat loaded using Chat. If a value is given, `chat` is ignored.
- **chat** (*Chat, optional*) – Chat data. Object obtained when chat loaded using Chat. Required if `df` is None.

Attributes

`user_color_mapping`	Get mapping between user and color.
`usernames`	Get list with users available in given chat.

Methods

`user_interventions_count_linechart`([...])	Plot number of user interventions over time.
`user_message_responses_flow`([title])	Get the flow of message responses.
`user_message_responses_heatmap`([norm, title])	Get the response matrix heatmap.
`user_msg_length_boxplot`([title, xlabel])	Generate figure with boxplots of each user's message length.

property user_color_mapping

Get mapping between user and color.

Each user is assigned a color automatically, so that this color is preserved for that user in all to-be-generated plots.

Returns *dict* – Mapping from username to color (rgb).

user_interventions_count_linechart(*date_mode='date'*, *msg_length=False*, *cumulative=False*, *all_users=False*, *title='User interventions count'*, *xlabel='Date/Time'*, *cummulative=None*)

Plot number of user interventions over time.

Parameters

- **date_mode** (*str, optional*) – Choose mode to group interventions by. Defaults to `'date'`. Available modes are:
  - `'date'`: Grouped by particular date (year, month and day).
  - `'hour'`: Grouped by hours.
  - `'month'`: Grouped by months.
  - `'weekday'`: Grouped by weekday (i.e. monday, tuesday, . . . , sunday).
  - `'hourweekday'`: Grouped by weekday and hour.
- **msg_length** (*bool, optional*) – Set to True to count the number of characters instead of number of messages sent.
- **cumulative** (*bool, optional*) – Set to True to obtain commulative counts.
- **all_users** (*bool, optional*) – Obtain number of interventions of all users combined. Defaults to False.

- **title** (*str, optional*) – Title for plot. Defaults to "User interventions count".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to "Date/Time".
- **cummulative** (*bool, optional*) – Deprecated, use cumulative.

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_interventions_count`
- `fig_scatter_time`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_interventions_count_
↪linechart(cumulative=True)
>>> plot(fig)
```

user_message_responses_flow (*title='Message flow'*)

Get the flow of message responses.

A response from user X to user Y happens if user X sends a message right after a message from user Y.

Uses a Sankey diagram.

Parameters **title** (*str, optional*) – Title for plot. Defaults to "Message flow".

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_response_matrix`
- `fig_sankey`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_message_responses_flow()
>>> plot(fig)
```

user_message_responses_heatmap (*norm='absolute', title='Response matrix'*)

Get the response matrix heatmap.

A response from user X to user Y happens if user X sends a message right after a message from user Y.

Parameters

- **norm** (*str, optional*) – Specifies the type of normalization used for reponse count. Can be:
  - `'absolute'`: Absolute count of messages.

- `'joint'`: Normalized by total number of messages sent by all users.
- `'sender'`: Normalized per sender by total number of messages sent by user.
- `'receiver'`: Normalized per receiver by total number of messages sent by user.

- **title** (*str, optional*) – Title for plot. Defaults to "Response matrix".

Returns *plotly.graph_objs.Figure* – Plotly Figure.

See also:

- `get_response_matrix`
- `fig_heatmap`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_message_responses_heatmap()
>>> plot(fig)
```

user_msg_length_boxplot (*title='User message length'*, *xlabel='User'*)

Generate figure with boxplots of each user's message length.

Parameters

- **title** (*str, optional*) – Title for plot. Defaults to "User message length".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to "User".

Returns *dict* – Dictionary with data and layout. Plotly compatible.

See also:

- `fig_boxplot_msglen`

Example

```
>>> from whatstk import WhatsAppChat
>>> from whatstk.graph import plot, FigureBuilder
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM)
>>> fig = FigureBuilder(chat=chat).user_msg_length_boxplot()
>>> plot(fig)
```

property usernames

Get list with users available in given chat.

Returns *list* – List with usernames available in chat DataFrame.

whatstk.graph.figures

Build Plotly compatible Figures.

whatstk.graph.figures.boxplot

Boxplot figures.

Functions

fig_boxplot_msglen(df[, ...])	username_to_color, Visualize boxplot.

whatstk.graph.figures.boxplot.**fig_boxplot_msglen**(*df*, *username_to_color=None*, *title=''*, *xlabel=None*)

Visualize boxplot.

Parameters

- **df** (*pandas.DataFrame*) – Chat data.
- **username_to_color** (*dict, optional*) –
- **title** (*str, optional*) – Title for plot. Defaults to "".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to None.

Returns plotly.graph_objs.Figure

whatstk.graph.figures.heatmap

Heatmap plot figures.

Functions

fig_heatmap(df_matrix[, title])	Generate heatmap figure from NxN matrix.

whatstk.graph.figures.heatmap.**fig_heatmap**(*df_matrix*, *title=''*)

Generate heatmap figure from NxN matrix.

Parameters

- **df_matrix** (*pandas.DataFrame*) – Matrix as DataFrame. Index values and column values must be equal.
- **title** (*str*) – Title of plot. Defaults to "".

Returns plotly.graph_objs.Figure

whatstk.graph.figures.sankey

Sankey plot figures.

Functions

fig_sankey(label, color, source, target, value)	Generate sankey image.

whatstk.graph.figures.sankey.**fig_sankey**(*label*, *color*, *source*, *target*, *value*, *title=''*)

Generate sankey image.

Parameters

- **label** (*list*) – List with node labels.
- **color** (*list*) – List with node colors.
- **source** (*list*) – List with link source id.
- **target** (*list*) – List with linke target id.
- **value** (*list*) – List with link value.
- **title** (*str, optional*) – Title. Defaults to "".

Returns plotly.graph_objs.Figure

whatstk.graph.figures.scatter

Scatter plot figures.

Functions

fig_scatter_time(user_data[, ...])	Obtain Figure to plot using plotly.

whatstk.graph.figures.scatter.**fig_scatter_time**(*user_data*, *username_to_color=None*, *title=''*, *xlabel=None*)

Obtain Figure to plot using plotly.

`user_data` must be a pandas.DataFrame with timestamps as index and a column for each user. You can easily generate suitable `user_data` using the function `get_interventions_count` (disclaimer: not compatible with `date_mode='hourweekday'`).

Parameters

- **user_data** (*pandas.DataFrame*) – Input data. Shape nrows x ncols, where nrows = number of timestaps and ncols = number of users.
- **username_to_color** (*dict, optional*) –
- **title** (*str, optional*) – Title of figure. Defaults to "".
- **xlabel** (*str, optional*) – x-axis label title. Defaults to None.

Returns plotly.graph_objs.Figure

See also:

- `get_interventions_count`

whatstk.graph.figures.utils

Utils for library plots.

Functions

hex_color_palette(n_colors)	Get palette of *n_colors* color hexadecimal codes.

whatstk.graph.figures.utils.**hex_color_palette**(*n_colors*)
Get palette of *n_colors* color hexadecimal codes.

Parameters **n_colors** (*int*) – Size of the color palette.

whatstk.utils

Library generic utils.

---

whatstk.utils.chat_merge

Merging chats.

Functions

merge_chats(dfs)	Merge several chats into a single one.

whatstk.utils.chat_merge.**merge_chats**(*dfs*)
Merge several chats into a single one.

Can come in handy when you have old exports and new ones, and both have relevant data.

Note: The dataframes must have an index with the timestamps of the messages, as this is required to correctly sort and merge the chats.

Parameters **dfs** (*List[pandas.DataFrame]*) – List with the chats as DataFrames.

Returns *pandas.DataFrame* – Merged chat.

---

whatstk.utils.exceptions

Library exceptions.

Exceptions

HFormatError	Raised when hformat could not be found.
RegexError	Raised when regex match is not possible.

exception whatstk.utils.exceptions.**HFormatError**
Bases: Exception

Raised when hformat could not be found.

exception whatstk.utils.exceptions.**RegexError**
Bases: Exception

Raised when regex match is not possible.

---

whatstk.utils.utils

Utils.

Classes

ColnamesDf	Access class constants using variable whatstk.utils.utils.COLNAMES_DF.

class whatstk.utils.utils.**ColnamesDf**
Bases: object

Access class constants using variable whatstk.utils.utils.COLNAMES_DF.

Example

Attributes

DATE	Date column
MESSAGE	Message column
MESSAGE_LENGTH	Message length column
USERNAME	Username column

Access constant COLNAMES_DF.DATE:

```
>>> from whatstk.utils.utils import COLNAMES_DF
>>> COLNAMES_DF.DATE
'date'
```

DATE = 'date'
Date column

MESSAGE = 'message'
Message column

MESSAGE_LENGTH = 'message_length'
Message length column

USERNAME = 'username'
Username column

whatstk.data

Load sample chats.

Tthis module contains the links to currently online-available chats. For more details, please refer to the source code.

Classes

`Urls`(POKEMON, LOREM, LOREM1, LOREM2, LOREM_2000)	

class whatstk.data.**Urls**(*POKEMON*, *LOREM*, *LOREM1*, *LOREM2*, *LOREM_2000*)

Bases: `tuple` **Attributes**

`LOREM`	Alias for field number 1
`LOREM1`	Alias for field number 2
`LOREM2`	Alias for field number 3
`LOREM_2000`	Alias for field number 4
`POKEMON`	Alias for field number 0

property LOREM
Alias for field number 1

property LOREM1
Alias for field number 2

property LOREM2
Alias for field number 3

property LOREM_2000
Alias for field number 4

property POKEMON
Alias for field number 0

whatstk._chat

Library objects.

Classes

BaseChat(df[, platform])	Base chat object.

class whatstk._chat.**BaseChat**(*df*, *platform=None*)

Bases: `object`

Base chat object. **Attributes**

df	Chat as DataFrame.
end_date	Chat end date.
start_date	Chat starting date.
users	List with users.

Methods

from_source(**kwargs)	Load chat.
merge(chat[, rename_users])	Merge current instance with `chat`.
rename_users(mapping)	Rename users.
to_csv(filepath)	Save chat as csv.

df
Chat as pandas.DataFrame.

See also:

- `WhatsAppChat`

property df
Chat as DataFrame.

Returns pandas.DataFrame

property end_date
Chat end date.

Returns datetime

classmethod from_source(***kwargs*)
Load chat.

Parameters **kwargs** – Specific to the child class.

Raises **NotImplementedError** – Must be implemented in children.

See also:

- `WhatsAppChat.from_source`

merge(*chat*, *rename_users=None*)
Merge current instance with `chat`.

Parameters

- **chat** (*WhatsAppChat*) – Another chat.
- **rename_users** (*dict*) – Dictionary mapping old names to new names. Example: {'John':['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *WhatsAppChat* – Merged chat.

See also:

- *rename_users*
- `merge_chats`

Example

Merging two chats can become handy when you have exported a chat in different times with your phone and hence each exported file might contain data that is unique to that file.

In this example however, we merge files from different chats.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> filepath_1 = whatsapp_urls.LOREM1
>>> filepath_2 = whatsapp_urls.LOREM2
>>> chat_1 = WhatsAppChat.from_source(filepath=filepath_1)
>>> chat_2 = WhatsAppChat.from_source(filepath=filepath_2)
>>> chat = chat_1.merge(chat_2)
```

rename_users (*mapping*)
Rename users.

This might be needed in multiple occations:

- Change typos in user names stored in phone.
- **If a user appears multiple times with different usernames, group these under the same name (this might** happen when multiple chats are merged).

Parameters **mapping** (*dict*) – Dictionary mapping old names to new names, example: {'John': ['Jon', 'J'], 'Ray': ['Raymond']} will map 'Jon' and 'J' to 'John', and 'Raymond' to 'Ray'. Note that old names must come as list (even if there is only one).

Returns *pandas.DataFrame* – DataFrame with users renamed according to *mapping*.

Raises **ValueError** – Raised if mapping is not correct.

Examples

Load LOREM2 chat and rename users *Maria* and *Maria2* to *Mary*.

```
>>> from whatstk.whatsapp.objects import WhatsAppChat
>>> from whatstk.data import whatsapp_urls
>>> chat = WhatsAppChat.from_source(filepath=whatsapp_urls.LOREM2)
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Maria', 'Maria2']
>>> chat = chat.rename_users(mapping={'Mary': ['Maria', 'Maria2']})
>>> chat.users
['+1 123 456 789', 'Giuseppe', 'John', 'Mary']
```

property start_date
Chat starting date.

Returns datetime

to_csv (*filepath*)
Save chat as csv.

Parameters **filepath** (*str*) – Name of file.

property users
List with users.

Returns list

5.4.3 Command line tools

whatstk-to-csv

Convert a WhatsApp txt file to csv.

```
usage: whatstk-to-csv [-h] [-f HFORMAT] input_filename output_filename

Convert a Whatsapp chat from csv to txt.

positional arguments:
input_filename        Input txt file.
output_filename       Name of output csv file.

optional arguments:
-h, --help            show this help message and exit
-f HFORMAT, --hformat HFORMAT
                        By default, auto-header detection isattempted. If does
                        not work, you can specify it manually using this
                        argument.
```

whatstk-graph

Get graph from your WhatsApp txt file.

```
usage: whatstk-graph [-h] [-o OUTPUT_FILENAME]
                     [-t {interventions_count,msg_length}]
                     [-id {date,hour,weekday,month}] [-ic] [-il] [-f HFORMAT]
                     input_filename

Visualise a WhatsApp chat. For advance settings, see package
librarydocumentation

positional arguments:
input_filename        Input txt file.

optional arguments:
-h, --help            show this help message and exit
-o OUTPUT_FILENAME, --output_filename OUTPUT_FILENAME
                        Graph generated can be stored as an HTMLfile.
-t {interventions_count,msg_length}, --type {interventions_count,msg_length}
                        Type of graph.
-id {date,hour,weekday,month}, --icount-date-mode {date,hour,weekday,month}
                        Select date mode. Only valid for
                        --type=interventions_count.
-ic, --icount-cumulative
                        Show values in a cumulative fashion. Only valid for
                        --type=interventions_count.
-il, --icount-msg-length
                        Count an intervention with its number of characters.
                        Otherwise an intervention is count as one.Only valid
                        for --type=interventions_count.
-f HFORMAT, --hformat HFORMAT
                        By default, auto-header detection isattempted. If does
                        not work, you can specify it manually using this
                        argument.
```

whatstk-generate-chat

Generate random WhatsApp chat.

```
whatstk-generate-chat --help
usage: Generate chat. [-h] -o OUTPUT_PATH
                      [--filenames FILENAMES [FILENAMES ...]] [-s SIZE]
                      [-f HFORMATS [HFORMATS ...]]
                      [--last-timestamp LAST_TIMESTAMP] [-v]

optional arguments:
-h, --help            show this help message and exit
-o OUTPUT_PATH, --output-path OUTPUT_PATH
                        Path where to store generated chats. Must exist.
--filenames FILENAMES [FILENAMES ...]
                        Filenames. Must be equal length of --hformats.
-s SIZE, --size SIZE  Number of messages to create per chat. Defaults to
                        500.
-f HFORMATS [HFORMATS ...], --hformats HFORMATS [HFORMATS ...]
                        Header format. If None, defaults to all supported
                        hformats. List formats as 'format 1' 'format 2' ...
--last-timestamp LAST_TIMESTAMP
                        Timestamp of last message. Format YYYY-mm-dd
-v, --verbose         Verbosity.
```

5.5 Why choose whatstk?

There are many python libraries to deal with WhatsApp and other platform chat files. Why should you choose **whatstk** over these?

5.5.1 Automatic parser

In WhatsApp, the chat might be exported in *different formats* depending on your phone configuration, which adds complexity when parsing the chat. **whatstk** incorporates a reliable and powerful `parser` to correctly infer the structure of most of the chats. In the rare and improbable case that the automatic parser does not work for a certain chat, you can still use hformat.

5.5.2 The power of pandas and plotly

whatstk uses well established and mantained python libraries pandas to process the data and plotly and exploits their potential to efficiently process and create figures.

5.5.3 Open source and Community oriented

The project is distributed under the GPL-3.0 license, available on GitHub and open for user contributions.

The project is mantained since 2016 by @lucasrodes.

5.6 Community & Governance

whatstk is a fully open-source project done for and by the community. It is primarily developed at sociepy by the whatstk team, with the help of open-source developers.

For library discussions, consider joining gitter group.

5.6.1 Leadership

BDFL

Role: final call in decisions related to the API.

- Lucas Rodés-Guirao

Community Contributors

- Albert Aparicio Isarn
- Kolmar Kafran
- Clara Sáez Calabuig (project logo)

5.7 Contribute

We are really open to your thoughts and feedback!

5.7.1 Bug reporting

Please report any bug that you may find to the issues section.

5.7.2 Requesting a Feature

If you find a new feature could be useful for the community, please try to add it in the issues section with a clear description.

5.7.3 Submitting a Pull Request

- Start by forking the develop branch.
- Add your code to the project!
- Test your code running script run-tests.sh.

This script checks the code style (flake8) and the logic of your code (pytest). Note: Make sure to open and read it. The first time you will need to run steps 1.1, 1.2 and 1.3.

```
sh ./run-tests.sh
```

This script generates three HTML files which are placed within a created folder *reports*.

- Once your code successfully passed the tests, you can submitt a pull request and wait for its aproval

Aproval of pull request

A pull request will be accepted if:

- Adds new functionalities of interest.
- Does not decrease the overall project code coverage.

Note: You will need to add tests for your code. For this, you can check the current tests.

5.7.4 Adding new examples

To add new examples, consider editing yourself a `rst` file in `docs/source/` directory in the repository. For questions or doubts, join the gitter group.

5.7.5 API discussions

Consider joining the gitter group.

5.7.6 Doubts?

Feel free to contact me :)

5.8 Changelog

5.8.1 Unreleased

- **Merge pull request #115 from lucasrodes/release/0.4.1** by *Lucas Rodés-Guirao* at *2021-04-28 21:55:54*

  DOC: Release v0.4.1
- **Bug fix** by *lucas rg* at *2021-04-28 21:46:21*
- **ENH: Export to csv without index column** by *lucas rg* at *2021-04-28 21:27:44*
- **0.4.1** by *lucas rg* at *2021-04-28 21:07:00*
- **Bump version: 0.4.1.dev0 → 0.4.2.dev0** by *lucas rg* at *2021-04-28 21:05:40*
- **dev** by *lucas rg* at *2021-04-28 21:05:25*
- **update requirements** by *lucas rg* at *2021-04-28 20:59:53*
- **Change changelog** by *lucas rg* at *2021-04-28 20:38:45*
- **fixed typos** by *lucas rg* at *2021-04-28 20:19:11*
- **Merge pull request #112 from lucasrodes/develop** by *Lucas Rodés-Guirao* at *2021-04-05 12:49:57*

  readme small fixes
- **readme small fixes** by *lucas rg* at *2021-04-04 16:27:12*
- **Merge pull request #111 from lucasrodes/develop** by *Lucas Rodés-Guirao* at *2021-04-04 16:15:46*

  Update links
- **Merge pull request #108 from lucasrodes/feature/badges** by *Lucas Rodés-Guirao* at *2021-03-11 22:08:39*

  Feature/badges
- **fix links** by *lucas rg* at *2021-03-11 19:53:42*
- **merge** by *lucas rg* at *2021-03-11 19:48:04*
- **citation** by *lucas rg* at *2021-01-26 19:11:11*
- **citation** by *lucas rg* at *2021-01-26 19:10:22*
- **Merge pull request #107 from lucasrodes/feature/readthedocs** by *Lucas Rodés-Guirao* at *2021-01-24 21:37:01*

  Feature/readthedocs
- **remove documentation generation** by *lucas rg* at *2021-01-24 16:18:09*
- **change doc URLs to read the docs** by *lucas rg* at *2021-01-24 16:02:34*
- **remove documentation creation from travis-ci. Using read the docs instead** by *lucas rg* at *2021-01-24 15:53:50*

5.8.2 v0.4.x

- **Merge pull request #106 from lucasrodes/release/0.4.0** by *Lucas Rodés-Guirao* at *2021-01-24 01:24:21*

  Release/0.4.0
- **pages changelog** by *lucas rg* at *2021-01-24 01:12:04*
- **Bump version: 0.4.0.rc0 → 0.4.0** by *lucas rg* at *2021-01-24 01:09:34*
- **Bump version: 0.4.0.b0 → 0.4.0.rc0** by *lucas rg* at *2021-01-24 01:09:32*
- **Bump version: 0.4.0.a0 → 0.4.0.b0** by *lucas rg* at *2021-01-24 01:09:29*
- **Bump version: 0.4.0.dev1 → 0.4.0.a0** by *lucas rg* at *2021-01-24 01:06:25*
- **bumping** by *lucas rg* at *2021-01-24 01:06:06*
- **travis ci test** by *lucas rg* at *2021-01-23 16:00:14*
- **travis CI test** by *lucas rg* at *2021-01-23 15:56:22*
- **testing travis CI** by *lucas rg* at *2021-01-23 15:52:55*
- **bumpversion 0.4.0.dev0** by *lucas rg* at *2021-01-23 15:38:55*
- **Merge pull request #105 from lucasrodes/feature/py39** by *Lucas Rodés-Guirao* at *2021-01-23 15:23:12*

  Feature/py39
- **travis trigger test** by *lucas rg* at *2021-01-23 15:17:37*
- **travis trigger test** by *lucas rg* at *2021-01-23 15:16:22*
- **travis not being triggered** by *lucas rg* at *2021-01-23 15:14:48*
- **Merge pull request #104 from lucasrodes/feature/py39** by *Lucas Rodés-Guirao* at *2021-01-23 15:10:45*

  Feature/py39
- **travis** by *lucas rg* at *2021-01-23 15:09:34*
- **requirement >py37** by *lucas rg* at *2021-01-23 14:31:50*
- **Merge pull request #103 from lucasrodes/feature/py39** by *Lucas Rodés-Guirao* at *2021-01-14 23:38:51*

  Add Python 3.9 support
- **remove 3.6, added 3.9** by *lucas rg* at *2021-01-14 21:46:47*
- **update requirements** by *lucas rg* at *2021-01-14 21:44:05*
- **3.9** by *lucas rg* at *2021-01-14 21:42:48*
- **add badge 3.9** by *lucas rg* at *2021-01-14 21:40:32*
- **ignore py39 env** by *lucas rg* at *2021-01-14 20:52:46*
- **Merge pull request #100 from lucasrodes/feature/change-df-index** by *Lucas Rodés-Guirao* at *2020-11-03 16:05:42*

  Feature/change df index
- **code style flake compliant** by *Lucas Rodes-Guirao* at *2020-11-02 22:19:33*
- **contribute section improved** by *Lucas Rodes-Guirao* at *2020-11-02 21:40:23*
- **removed unnecessary file** by *Lucas Rodes-Guirao* at *2020-11-02 21:19:40*
- **bugfix** by *Lucas Rodes-Guirao* at *2020-11-02 21:19:28*

- **doc updates** by *Lucas Rodes-Guirao* at *2020-11-02 21:18:57*
- **docs** by *Lucas Rodes-Guirao* at *2020-11-02 00:08:15*
- **chat generation updates, now generates numeric index and date as column** by *Lucas Rodes-Guirao* at *2020-11-02 00:07:52*
- **merge util now using column and not index** by *Lucas Rodes-Guirao* at *2020-11-01 23:54:37*
- **adequate tests for new df structure** by *Lucas Rodes-Guirao* at *2020-11-01 23:54:16*
- **interventions module now fully funcional** by *Lucas Rodes-Guirao* at *2020-11-01 23:42:59*
- **removed duplicate call to function** by *Lucas Rodes-Guirao* at *2020-11-01 23:03:45*
- **Merge pull request #98 from lucasrodes/feature/requirements-update** by *Lucas Rodés-Guirao* at *2020-11-01 22:40:50*

  Feature/requirements update
- **library versions updated** by *Lucas Rodes-Guirao* at *2020-11-01 21:50:04*
- **Merge pull request #97 from lucasrodes/develop** by *Lucas Rodés-Guirao* at *2020-08-09 14:50:58*

  Develop
- **Merge pull request #96 from lucasrodes/feature/hit-badge-fix** by *Lucas Rodés-Guirao* at *2020-08-09 14:30:59*

  hits extension fixed, using wesky93/views solution, based on hits
- **hits extension fixed, using wesky93/views solution, based on hits** by *lucas rg* at *2020-08-09 14:17:10*
- **Merge pull request #95 from lucasrodes/develop** by *Lucas Rodés-Guirao* at *2020-08-08 19:37:04*

  Develop
- **Merge pull request #94 from lucasrodes/feature/badges** by *Lucas Rodés-Guirao* at *2020-08-08 19:30:23*

  badges
- **badges** by *lucas rg* at *2020-08-08 19:24:18*
- **Create FUNDING.yml** by *Lucas Rodés-Guirao* at *2020-07-30 11:35:08*

5.8.3 v0.3.x

- **Merge pull request #89 from lucasrodes/release/0.3.0** by *Lucas Rodés-Guirao* at *2020-06-26 21:53:41*

  bug in test
- **bug in test** by *lucas rg* at *2020-06-26 21:27:51*
- **Merge pull request #88 from lucasrodes/release/0.3.0** by *Lucas Rodés-Guirao* at *2020-06-26 21:22:43*

  Release/0.3.0
- **Bump version: 0.3.0.rc2 → 0.3.0** by *lucas rg* at *2020-06-26 21:10:40*
- **transitioning version** by *lucas rg* at *2020-06-26 21:10:00*
- **Bump version: 0.3.0.rc0 → 0.3.0.rc1** by *lucas rg* at *2020-06-26 12:14:48*
- **typo in classifier** by *lucas rg* at *2020-06-26 12:14:45*
- **Bump version: 0.3.0.b3 → 0.3.0.rc0** by *lucas rg* at *2020-06-26 12:05:38*

- **Merge pull request #87 from lucasrodes/feature/travis-pages-deploy-test** by *Lucas Rodés-Guirao* at *2020-06-26 12:04:12*

  Feature/travis pages deploy test

- **deploy to pypi activated** by *lucas rg* at *2020-06-26 11:58:20*
- **setup updated, new classifiers added** by *lucas rg* at *2020-06-26 11:57:42*
- **Bump version: 0.3.0.b2 → 0.3.0.b3** by *lucas rg* at *2020-06-26 11:36:38*
- **ignore gitignore** by *lucas rg* at *2020-06-26 11:31:31*
- **changelog file as markdown** by *lucas rg* at *2020-06-26 11:30:34*
- **Bump version: 0.3.0.b1 → 0.3.0.b2** by *lucas rg* at *2020-06-26 10:19:40*
- **links fixed** by *lucas rg* at *2020-06-26 10:19:16*
- **removed build files** by *lucas rg* at *2020-06-26 09:58:14*
- **Bump version: 0.3.0.b0 → 0.3.0.b1** by *lucas rg* at *2020-06-26 09:35:47*
- **revert gitignore changes** by *lucas rg* at *2020-06-26 09:32:28*
- **reorganization** by *lucas rg* at *2020-06-26 09:26:11*
- **testing pages deployment** by *lucas rg* at *2020-06-26 09:22:41*
- **reorganized docs structure** by *lucas rg* at *2020-06-26 09:22:28*
- **travis wip** by *lucas rg* at *2020-06-25 21:45:25*
- **wip travis** by *lucas rg* at *2020-06-25 20:23:57*
- **Merge pull request #86 from lucasrodes/feature/documentation** by *Lucas Rodés-Guirao* at *2020-06-25 20:10:25*

  Feature/documentation

- **going to beta version** by *lucas rg* at *2020-06-25 20:03:42*
- **testing travis** by *lucas rg* at *2020-06-25 20:00:23*
- **Merge pull request #85 from lucasrodes/feature/documentation** by *Lucas Rodés-Guirao* at *2020-06-25 19:35:13*

  fixed link to documentation

- **fixed link to documentation** by *lucas rg* at *2020-06-25 19:33:41*
- **Merge pull request #84 from lucasrodes/feature/documentation** by *Lucas Rodés-Guirao* at *2020-06-25 19:30:15*

  Feature/documentation

- **removed tests in travis pipeline for 3.5** by *lucas rg* at *2020-06-25 19:23:31*
- **changed minimum version to 3.6** by *lucas rg* at *2020-06-25 19:22:10*
- **removed version 3.9** by *lucas rg* at *2020-06-25 19:19:59*
- **travis & version requirements changed to 3.5** by *lucas rg* at *2020-06-25 19:18:04*
- **travis** by *lucas rg* at *2020-06-25 19:13:50*
- **travis** by *lucas rg* at *2020-06-25 18:59:32*
- **travis** by *lucas rg* at *2020-06-25 18:23:14*

- **test travis** by *lucas rg* at *2020-06-25 18:19:01*
- **remove unused extenstion sphinx_multiversion (2)** by *lucas rg* at *2020-06-25 17:56:42*
- **remove unused extenstion sphinx_multiversion** by *lucas rg* at *2020-06-25 17:54:43*
- **docs dependencies installation added to travis config file** by *lucas rg* at *2020-06-25 17:50:08*
- **working on stages, ideally: test in multiple environments, deploy only in one** by *lucas rg* at *2020-06-25 17:38:13*
- **added Clara Saez Calabuig as the logo designer** by *lucas rg* at *2020-06-25 17:12:50*
- **changed urls for sample chats, now using develop** by *lucas rg* at *2020-06-25 17:11:59*
- **Merge pull request #83 from lucasrodes/feature/documentation** by *Lucas Rodés-Guirao* at *2020-06-25 17:10:48*

  Feature/documentation
- **end line break** by *lucas rg* at *2020-06-25 17:02:01*
- **links corrected** by *lucas rg* at *2020-06-25 17:01:31*
- **docs update. Travis now pushing updated documentation automatically to gh-pages (in beta)** by *lucas rg* at *2020-06-25 17:01:19*
- **fixed flake** by *lucas rg* at *2020-06-24 22:07:18*
- **version update** by *lucas rg* at *2020-06-24 21:57:27*
- **docstring update** by *lucas rg* at *2020-06-24 21:54:14*
- **fixed cmd api documentation** by *lucas rg* at *2020-06-24 21:53:38*
- **merge chat examples changed in docstring** by *lucas rg* at *2020-06-24 21:52:57*
- **update library script to generate chats** by *lucas rg* at *2020-06-24 21:51:01*
- **docs updated, typos fixed, added few new files.** by *lucas rg* at *2020-06-24 21:50:35*
- **new custom plot example added** by *lucas rg* at *2020-06-23 17:34:10*
- **update docs** by *lucas rg* at *2020-06-23 16:27:30*
- **gitignore doctrees** by *lucas rg* at *2020-06-19 18:01:59*
- **gitignore updated** by *lucas rg* at *2020-06-18 19:51:04*
- **documentation doctrees** by *lucas rg* at *2020-06-18 19:50:46*
- **htmls** by *lucas rg* at *2020-06-18 19:50:14*
- **commented future library to use multiple version self-hosted** by *lucas rg* at *2020-06-18 19:35:02*
- **docs update, whatsappchat object docstring** by *lucas rg* at *2020-06-18 18:54:10*
- **improved norm docstring** by *lucas rg* at *2020-06-17 21:43:05*
- **added norm argument in figurebuilder plot method** by *lucas rg* at *2020-06-17 21:42:56*
- **docs update** by *lucas rg* at *2020-06-17 21:42:16*
- **examples and code adapted for #82** by *lucas rg* at *2020-06-16 16:33:45*
- **bug in test 2** by *lucas rg* at *2020-06-16 16:27:18*
- **bug in test** by *lucas rg* at *2020-06-16 16:25:12*
- **testing from_source in BaseChat, all_users in FigureBuilder** by *lucas rg* at *2020-06-16 16:24:31*

- **working on visualisation examples** by *lucas rg* at *2020-06-16 16:20:07*
- **docs update** by *lucas rg* at *2020-06-16 15:22:07*
- **figure methods returning go.Figure objects** by *lucas rg* at *2020-06-16 13:52:56*
- **example plots** by *lucas rg* at *2020-06-14 16:06:50*
- **refactor docs, removed developer guide and moved its content to getting started section. added info on library available chats** by *lucas rg* at *2020-06-14 16:06:37*
- **source update** by *lucas rg* at *2020-06-14 14:08:25*
- **modified chat so it contains same user with different names** by *lucas rg* at *2020-06-14 14:08:06*
- **minor erorrs in docstring** by *lucas rg* at *2020-06-14 14:03:29*
- **examples** by *lucas rg* at *2020-06-14 13:12:21*
- **boxplot example** by *lucas rg* at *2020-06-14 10:18:18*
- **link to lorem 2k** by *lucas rg* at *2020-06-14 00:11:15*
- **docs** by *lucas rg* at *2020-06-14 00:08:25*
- **docs4** by *lucas rg* at *2020-06-13 23:44:53*
- **Merge pull request #81 from lucasrodes/feature/sample-chat-in-library** by *Lucas Rodés-Guirao* at *2020-06-10 16:48:57*

  Feature/sample chat in library
- **changed url with files so it uses develop branch instead** by *lucas rg* at *2020-06-10 16:43:30*
- **note on future filepath_url** by *lucas rg* at *2020-06-10 16:39:13*
- **update examples chats folder, added lorem.txt example. refactor of whatstk.data accordingly** by *lucas rg* at *2020-06-10 16:38:08*
- **can read chats from URLs** by *lucas rg* at *2020-06-10 16:12:07*
- **Merge pull request #78 from lucasrodes/feature/refactor-v3** by *Lucas Rodés-Guirao* at *2020-06-10 15:00:13*

  Feature/refactor v3
- **removed WhatsAppChat from __init__** by *lucas rg* at *2020-06-10 14:49:22*
- **parser** by *lucas rg* at *2020-06-10 14:44:27*
- **changed place where HFormatError exception is raised** by *lucas rg* at *2020-06-10 14:44:04*
- **documentation** by *lucas rg* at *2020-06-10 14:43:46*
- **update imports in scripts** by *lucas rg* at *2020-06-09 22:34:28*
- **test renaming** by *lucas rg* at *2020-06-09 22:33:15*
- **change in chat generator** by *lucas rg* at *2020-06-09 22:31:45*
- **doc templates** by *lucas rg* at *2020-06-09 21:52:47*
- **reorganizing package, so in the future might incorporate parsers for other social networks (e.g. facebook). tests changed accordingly. documentation ongoing process** by *lucas rg* at *2020-06-09 21:52:19*
- **parser operations moved to module whatstk.parser** by *lucas rg* at *2020-06-09 17:10:50*
- **moved figurebuilder** by *lucas rg* at *2020-06-06 11:15:49*
- **update prenum** by *lucas rg* at *2020-06-04 16:48:11*

- **changed versioning from major.mninor.patchrelease to major.minor.patch.release** by *lucas rg* at *2020-06-04 16:46:57*
- **docs requirements** by *lucas rg* at *2020-06-04 16:46:31*
- **doc generation script** by *lucas rg* at *2020-06-04 16:46:16*
- **added new test** by *lucas rg* at *2020-06-04 16:45:17*
- **switch to sphinx** by *lucas rg* at *2020-06-04 16:44:55*
- **docs** by *lucas rg* at *2020-06-04 16:44:28*
- **requires python 3.7** by *lucas rg* at *2020-06-02 10:26:33*
- **setter for color user mapping** by *lucas rg* at *2020-06-01 23:54:50*
- **Clarified error message** by *lucas rg* at *2020-06-01 19:04:15*
- **image path change so it can be displayed on pypi** by *lucas rg* at *2020-06-01 17:06:27*
- **bugfix in docs** by *lucas rg* at *2020-06-01 16:58:44*
- **Merge pull request #76 from lucasrodes/feature/documentation** by *Lucas Rodés-Guirao* at *2020-06-01 16:55:37*

  Feature/documentation
- **fixed bug in module import** by *lucas rg* at *2020-06-01 16:51:23*
- **bumpversion file update** by *lucas rg* at *2020-06-01 16:44:44*
- **bumped dev version** by *lucas rg* at *2020-06-01 16:44:26*
- **bumpversion support for pre and prenum** by *lucas rg* at *2020-06-01 16:43:30*
- **added pre and prenum to bumpversion** by *lucas rg* at *2020-06-01 16:30:21*
- **bug** by *lucas rg* at *2020-06-01 15:41:15*
- **test version** by *lucas rg* at *2020-06-01 15:36:54*
- **removed conf.py file** by *lucas rg* at *2020-06-01 15:30:09*
- **docs** by *lucas rg* at *2020-06-01 15:23:14*
- **folder structure** by *lucas rg* at *2020-06-01 15:20:48*
- **docs mvp** by *lucas rg* at *2020-06-01 15:18:58*
- **documentation mvp** by *lucas rg* at *2020-06-01 15:17:43*
- **Merge branch 'develop' into feature/documentation** by *lucas rg* at *2020-05-31 23:47:57*
- **Merge pull request #75 from lucasrodes/feature/response-matrix** by *Lucas Rodés-Guirao* at *2020-05-31 23:47:30*

  Feature/response matrix
- **tests for new graphs** by *lucas rg* at *2020-05-31 23:42:13*
- **response matrix visualization, #20** by *lucas rg* at *2020-05-31 23:38:54*
- **added more files** by *lucas rg* at *2020-05-31 23:05:18*
- **sankey diagram to plot responses between used, #20** by *lucas rg* at *2020-05-31 23:04:48*
- **removed buggy examples** by *lucas rg* at *2020-05-31 23:04:27*
- **response_matrix moved to import form __init__** by *lucas rg* at *2020-05-31 23:03:54*

- **changed name from whatstk.plotly to whatstk.graph** by *lucas rg* at *2020-05-31 20:10:52*
- **response_matrix method added, #20** by *lucas rg* at *2020-05-31 18:24:50*
- **bug in _get_df, third useless argument** by *lucas rg* at *2020-05-31 18:23:16*
- **xml coverage report** by *lucas rg* at *2020-05-31 18:22:50*
- **mitigating hardcoding. now check if is df or chat is done in utils** by *lucas rg* at *2020-05-31 12:08:59*
- **merge with develop** by *lucas rg* at *2020-05-31 00:35:12*
- **Merge pull request #74 from lucasrodes/feature/plot-user-colors** by *Lucas Rodés-Guirao* at *2020-05-31 00:27:31*

  Feature/plot user colors
- **logo** by *lucas rg* at *2020-05-31 00:22:40*
- **logo** by *lucas rg* at *2020-05-31 00:22:32*
- **method to get n hexadecimal color codes** by *lucas rg* at *2020-05-31 00:18:38*
- **colors added using hsl seaborn palette** by *lucas rg* at *2020-05-31 00:18:13*
- **fix bug in vis method** by *lucas rg* at *2020-05-31 00:08:48*
- **method to generate n hexadecimal color codes** by *lucas rg* at *2020-05-30 23:49:03*
- **to US english** by *lucas rg* at *2020-05-30 23:48:31*
- **color generated per user** by *lucas rg* at *2020-05-30 23:48:12*
- **changed to US english** by *lucas rg* at *2020-05-30 23:47:26*
- **seaborn now is a requirement** by *lucas rg* at *2020-05-30 23:06:01*
- **Merge branch 'develop' into feature/plot-user-colors** by *lucas rg* at *2020-05-30 22:35:08*
- **Merge pull request #73 from lucasrodes/feature/chat-df-schema** by *Lucas Rodés-Guirao* at *2020-05-30 22:33:30*

  Feature/chat df schema
- **removed debugging prints** by *lucas rg* at *2020-05-30 22:25:38*
- **grouped colnames naming of chat df under whatstk.utils.utils and added schema to df (now using string instead of object, pandas>=1.0.0)** by *lucas rg* at *2020-05-30 22:25:21*
- **grouped colnames naming of chat df under whatstk.utils.utils** by *lucas rg* at *2020-05-30 22:24:06*
- **get users** by *lucas rg* at *2020-05-30 17:19:44*
- **Merge pull request #71 from lucasrodes/feature/cmd-visualise** by *Lucas Rodés-Guirao* at *2020-05-30 16:19:17*

  Feature/cmd visualise
- **command line tool to generate graphs** by *lucas rg* at *2020-05-30 16:04:24*
- **refactor input argument names** by *lucas rg* at *2020-05-30 16:04:08*
- **spacing** by *lucas rg* at *2020-05-30 14:53:39*
- **bug in hformat** by *lucas rg* at *2020-05-30 14:53:19*
- **changed readme title** by *lucas rg* at *2020-05-30 14:29:43*

- **Merge pull request #70 from lucasrodes/feature/plotly-module** by *Lucas Rodés-Guirao* at *2020-05-30 14:22:11*

  Feature/plotly module

- **changed description** by *lucas rg* at *2020-05-30 01:05:10*
- **plotly module** by *lucas rg* at *2020-05-30 00:44:19*
- **Merge branch 'feature/plotly-module' into feature/documentation** by *lucas rg* at *2020-05-30 00:42:47*
- **added tests for plotly module, placed flake8 check before unittests** by *lucas rg* at *2020-05-30 00:41:38*
- **documentation** by *lucas rg* at *2020-05-29 22:29:47*
- **refactor** by *lucas rg* at *2020-05-29 18:59:01*
- **command line** by *lucas rg* at *2020-05-23 02:14:52*
- **readme ref** by *lucas rg* at *2020-05-23 02:12:04*
- **plot refactor** by *lucas rg* at *2020-05-23 02:07:24*
- **README** by *lucas rg* at *2020-05-23 02:03:15*
- **readme** by *lucas rg* at *2020-05-23 02:02:01*
- **readme** by *lucas rg* at *2020-05-23 02:01:00*
- **working on message length plot** by *lucas rg* at *2020-05-20 22:58:00*
- **added command line script to convert txt to csv. #65** by *lucas rg* at *2020-05-20 22:20:55*
- **Merge pull request #64 from lucasrodes/feature/test-minor-refactor** by *Lucas Rodés-Guirao* at *2020-05-20 20:48:13*

  Feature/test minor refactor

- **pull requests section change in README** by *lucas rg* at *2020-05-20 20:29:46*
- **line length changed** by *lucas rg* at *2020-05-20 17:48:22*
- **travis flake8 added** by *lucas rg* at *2020-05-20 17:44:49*
- **run test script** by *lucas rg* at *2020-05-20 16:01:09*
- **now using flake** by *lucas rg* at *2020-05-20 16:00:52*
- **test instructions** by *lucas rg* at *2020-05-20 16:00:27*
- **chat merge test** by *lucas rg* at *2020-05-20 16:00:16*
- **refactor chat merge util fie** by *lucas rg* at *2020-05-20 15:59:54*
- **Merge pull request #63 from lucasrodes/feature/merge-two-chats** by *Lucas Rodés-Guirao* at *2020-05-11 23:19:52*

  Feature/merge two chats

- **minor bugs** by *lucas rg* at *2020-05-11 23:14:29*
- **bug in merge_chats** by *lucas rg* at *2020-05-11 22:40:44*
- **bug in merge_chats** by *lucas rg* at *2020-05-11 22:40:22*
- **Merge pull request #60 from lucasrodes/feature/merge-two-chats** by *Lucas Rodés-Guirao* at *2020-05-11 22:22:15*

  Feature/merge two chats. Closes and Fixes #59 #41

- **added df_from_multiple_txt as root method, in whatstk** by *lucas rg* at *2020-05-11 22:13:59*
- **added df_from_multiple_txt method to whatstk** by *lucas rg* at *2020-05-11 22:13:15*
- **second part previous commit** by *lucas rg* at *2020-05-11 21:54:06*
- **merging two chats, and renaming users is now possible** by *lucas rg* at *2020-05-11 21:53:54*
- **command line script updated, now accepts output file name, and specific hformat list** by *lucas rg* at *2020-05-11 21:53:34*
- **tests done** by *lucas rg* at *2020-05-11 21:52:55*
- **example tests** by *lucas rg* at *2020-05-11 21:51:16*
- **Merge pull request #58 from lucasrodes/feature/df-integration-api** by *Lucas Rodés-Guirao* at *2020-05-10 23:26:11*

  Feature/df integration api
- **bug in tests** by *lucas rg* at *2020-05-10 23:24:00*
- **update docs, dealing with #52** by *lucas rg* at *2020-05-10 23:15:27*
- **Merge branch 'develop' into feature/df-integration-api** by *lucas rg* at *2020-05-10 23:12:27*
- **changin API, now supporting df when chat was the only option** by *lucas rg* at *2020-05-10 23:11:41*
- **Merge pull request #57 from lucasrodes/feature/test-examples** by *Lucas Rodés-Guirao* at *2020-05-10 22:58:57*

  Feature/test examples
- **close #56** by *lucas rg* at *2020-05-10 22:52:50*
- **mkdir for chats** by *lucas rg* at *2020-05-10 22:47:19*
- **travis build config** by *lucas rg* at *2020-05-10 22:38:03*
- **removed scripts from ignored folders** by *lucas rg* at *2020-05-10 22:34:11*
- **created exacutable to generate chats** by *lucas rg* at *2020-05-10 22:33:57*
- **towards usage of %y instead of %Y** by *lucas rg* at *2020-05-10 22:04:36*
- **going towards %y instead of %Y** by *lucas rg* at *2020-05-10 22:02:59*
- **commented unused methods** by *lucas rg* at *2020-05-10 22:02:33*
- **bug in to_txt argument default value** by *lucas rg* at *2020-05-10 22:02:15*
- **tests updated** by *lucas rg* at *2020-05-10 22:01:52*
- **refactor of ChatGenerator. export() deprecated** by *lucas rg* at *2020-05-10 21:29:24*
- **remove export test, method deprecated** by *lucas rg* at *2020-05-10 21:28:12*
- **fixed bug with PM/AM formats** by *lucas rg* at *2020-05-10 19:35:57*
- **tests/chats not needed** by *lucas rg* at *2020-05-10 19:34:51*
- **example chat removed** by *lucas rg* at *2020-05-10 19:34:31*
- **typo in arguments** by *lucas rg* at *2020-05-10 19:01:07*
- **renaming of methos** by *lucas rg* at *2020-05-10 18:58:52*
- **new method, generate_chats_hformats to export chats to given list of hformats (defaults to all supported formats)** by *lucas rg* at *2020-05-10 18:58:26*

- **minor changes in docstrings, using %y instead of %Y. Added username as phone number** by *lucas rg* at *2020-05-10 18:36:10*
- **auto_header big refactor. most changes reside in method _extract_header_parts().** by *lucas rg* at *2020-05-10 18:35:31*
- **added library exceptions** by *lucas rg* at *2020-05-10 18:34:01*
- **added new date codes for regex mapping. Added exception in parse_chat, when regex did not match any part of the input text** by *lucas rg* at *2020-05-10 18:33:38*
- **header support. All listed headers support auto-header** by *lucas rg* at *2020-05-10 18:28:58*
- **Merge pull request #51 from lucasrodes/feature/df-from-txt** by *Lucas Rodés-Guirao* at *2020-05-07 09:09:57*

  Feature/df from txt
- **modified examples** by *lucas rg* at *2020-05-07 09:06:47*
- **new method df_from_txt** by *lucas rg* at *2020-05-07 08:59:37*
- **Merge pull request #50 from lucasrodes/hotfix/TOC** by *Lucas Rodés-Guirao* at *2020-05-06 23:18:39*

  removed useless section from readme
- **removed useless section from readme** by *lucas rg* at *2020-05-06 23:16:25*

5.8.4 v0.2.x

- **Merge pull request #33 from lucasrodes/develop** by *Lucas Rodés-Guirao* at *2020-04-29 22:05:06*

  Develop
- **update minor 0.2.0** by *lucas rg* at *2020-04-29 21:59:55*
- **Merge pull request #32 from lucasrodes/feature/tests** by *Lucas Rodés-Guirao* at *2020-04-29 21:54:57*

  Feature/tests
- **readme badges python versio update** by *lucas rg* at *2020-04-29 21:49:26*
- **travis update** by *lucas rg* at *2020-04-29 21:46:02*
- **travis** by *lucas rg* at *2020-04-29 21:41:51*
- **wip travis** by *lucas rg* at *2020-04-29 21:38:55*
- **tests integrated** by *lucas rg* at *2020-04-29 21:28:49*
- **small fix in coloring badges** by *lucas rg* at *2020-04-29 20:44:55*
- **Merge pull request #31 from lucasrodes/feature/docs** by *Lucas Rodés-Guirao* at *2020-04-29 20:32:52*

  Feature/docs
- **finished docs** by *lucas rg* at *2020-04-29 20:18:34*
- **doc making script** by *lucas rg* at *2020-04-29 20:12:45*
- **script for docs** by *lucas rg* at *2020-04-29 20:12:06*
- **script for docs** by *lucas rg* at *2020-04-29 20:09:03*
- **pydoc markdown** by *lucas rg* at *2020-04-29 20:07:06*
- **plot module** by *lucas rg* at *2020-04-29 20:06:52*
- **refactor, new hierarchy and documentation passed to Google style** by *lucas rg* at *2020-04-29 18:50:44*

- **merge** by *lucas rg* at *2020-04-29 17:34:04*
- **Merge pull request #30 from lucasrodes/feature/header-auto-detect** by *Lucas Rodés-Guirao* at *2020-04-29 17:31:03*

  Feature/header auto detect
- **readme links** by *lucas rg* at *2020-04-29 17:28:53*
- **readme** by *lucas rg* at *2020-04-29 17:27:16*
- **update readme** by *lucas rg* at *2020-04-29 17:21:27*
- **wip readme** by *lucas rg* at *2020-04-29 16:44:19*
- **ignore file update** by *lucas rg* at *2020-04-29 16:27:56*
- **smodule comment** by *lucas rg* at *2020-04-29 16:26:30*
- **Merge branch 'master' into develop** by *lucas rg* at *2020-04-29 11:30:16*
- **removed useless space** by *lucas rg* at *2020-04-29 11:29:16*
- **backwards compatibility** by *lucas rg* at *2020-04-29 11:28:59*
- **small typo in imports** by *lucas rg* at *2020-04-29 11:28:49*
- **interventions and WhatsAppChat objects moved to __init__** by *lucas rg* at *2020-04-29 11:28:21*
- **readme instructions update, notes added** by *lucas rg* at *2020-04-29 11:27:58*
- **changed print to log** by *lucas rg* at *2020-04-29 11:27:39*
- **renamed whatstk.alpha to whatstk.utils** by *lucas rg* at *2020-04-29 11:13:21*
- **change for backwards compatibility** by *lucas rg* at *2020-04-29 10:59:25*
- **added header auto extraction when from_txt in WhatsAppChat object** by *lucas rg* at *2020-04-29 10:35:02*
- **Merge pull request #29 from lucasrodes/hotfix/link-to-gui-app** by *Lucas Rodés-Guirao* at *2020-04-25 17:58:49*

  changed header levels, add link to app
- **changed header levels, add link to app** by *lucas rg* at *2020-04-25 17:53:41*
- **import syntax changed** by *lucas rg* at *2020-04-25 17:50:22*
- **merge** by *lucas rg* at *2019-06-21 19:26:23*
- **regex update** by *lucas rg* at *2019-06-21 19:24:58*

PYTHON MODULE INDEX

W

INDEX

P

R

S

T

U

W