Hate Tweet Map

Release 1.0

Dario Amoroso d'Aragona

Jul 06, 2021

CONTENTS:

1 Code Documentation **1**
- 1.1 Tweets Process . . . 1
  - 1.1.1 TagMe Service . . . 1
  - 1.1.2 Entity Linker Service . . . 3
  - 1.1.3 Tweet Processor . . . 3
- 1.2 Tweets Searcher . . . 5
  - 1.2.1 Search Tweets . . . 5
- 1.3 Users Searcher . . . 6
  - 1.3.1 Search Users . . . 6
- 1.4 Manage Tweets . . . 7
  - 1.4.1 Manage Tweets . . . 7

2 User Guide **9**
- 2.1 Initialization . . . 9
  - 2.1.1 Setup . . . 9
- 2.2 Scripts . . . 9
  - 2.2.1 Search Tweets Script . . . 9
    - 2.2.1.1 Configuration file . . . 9
      - 2.2.1.1.1 Mandatory Section . . . 11
      - 2.2.1.1.2 Optional Section . . . 11
        - 2.2.1.1.2.1 1. Lang . . . 13
        - 2.2.1.1.2.2 2. Context Annotation . . . 13
        - 2.2.1.1.2.3 3. Number of results . . . 13
        - 2.2.1.1.2.4 4. Reach all tweets . . . 14
        - 2.2.1.1.2.5 5. Time . . . 14
        - 2.2.1.1.2.6 6. Geo . . . 15
        - 2.2.1.1.2.7 7. Filter retweet . . . 17
    - 2.2.1.2 Use the script . . . 17
  - 2.2.2 Search Users Script . . . 17
    - 2.2.2.1 Configuration file . . . 17
      - 2.2.2.1.1 Mongodb tweets . . . 18
      - 2.2.2.1.2 Mongodb Users . . . 18
      - 2.2.2.1.3 Twitter . . . 18
    - 2.2.2.2 Use the script . . . 18
  - 2.2.3 Process Tweets Script . . . 19
    - 2.2.3.1 Configuration file . . . 20
      - 2.2.3.1.1 Mongodb . . . 20
      - 2.2.3.1.2 Analyses:Nlp . . . 20
      - 2.2.3.1.3 Analyses:TagMe . . . 21
      - 2.2.3.1.4 Analyses:Sentiment Analyses . . . 21

2.2.3.1.5 Analyses:Geocoding . . . 22
2.2.3.1.6 Analyses:Analyze all tweets . . . 22
2.2.3.2 Use the script . . . 22
2.2.4 Manage Tweets Script . . . 22
2.2.4.1 Configuration file . . . 23
2.2.4.1.1 Mongodb . . . 23
2.2.4.1.2 Mode . . . 23
2.2.4.1.2.1 Mode: extract . . . 24
2.2.4.1.2.2 Mode: delete . . . 24
2.2.4.1.3 Criteria . . . 24
2.2.4.2 Use the script . . . 26

3 Indices and tables **27**

Python Module Index **29**

Index **31**

CHAPTER

ONE

CODE DOCUMENTATION

1.1 Tweets Process

1.1.1 TagMe Service

This module provides a wrapper for the TagMe API.

class hate_tweet_map.tweets_processor.MyTagMe.**AnnotateResponse**(*json_content*)
A response to a call to the annotation (/tag) service. It contains the list of annotations found.

get_annotations(*min_rho=None*)
Get the list of annotations found.

Parameters **min_rho** – if set, only get entities with a rho-score (confidence) higher than this.

class hate_tweet_map.tweets_processor.MyTagMe.**Annotation**(*ann_json*)
An annotation, i.e. a link of a part of text to an entity.

uri(*lang='en'*)
Get the URI of this annotation entity.

Parameters **lang** – the Wikipedia language.

class hate_tweet_map.tweets_processor.MyTagMe.**Mention**(*mention_json*)
A mention, i.e. a part of text that may mention an entity.

class hate_tweet_map.tweets_processor.MyTagMe.**MentionsResponse**(*json_content*)
A response to a call to the mention finding (/spot) service. It contains the list of mentions found.

get_mentions(*min_lp=None*)
Get the list of mentions found.

Parameters **min_lp** – if set, only get mentions with a link probability higher than this.

class hate_tweet_map.tweets_processor.MyTagMe.**Relatedness**(*rel_json*)
A relatedness, i.e. a real value between 0 and 1 indicating how semantically close two entities are.

as_pair()
Get this relatedness value as a pair (titles, rel), where rel is the relatedness value and titles is the pair of the two titles/Wikipedia IDs.

class hate_tweet_map.tweets_processor.MyTagMe.**RelatednessResponse**(*json_contents*)
A response to a call to the relatedness (/rel) service. It contains the list of relatedness for each pair.

get_relatedness(*i=0*)
Get the relatedness of a pairs of entities.

Parameters **i** – the index of an entity pair. The order is the same as the request.

hate_tweet_map.tweets_processor.MyTagMe.**annotate**(*text*, *is_twitter_text=False*, *gcube_token=None*, *lang='en'*, *api='https://tagme.d4science.org/tagme/tag'*, *long_text=3*)

Annotate a text, linking it to Wikipedia entities.

Parameters

- **is_twitter_text** – true if the text is a tweet, in this case it's possible pass the json as text parameter
- **text** – the text to annotate.
- **gcube_token** – the authentication token provided by the D4Science infrastructure.
- **lang** – the Wikipedia language.
- **api** – the API endpoint.
- **long_text** – long_text parameter (see TagMe documentation).

hate_tweet_map.tweets_processor.MyTagMe.**mentions**(*text*, *gcube_token=None*, *lang='en'*, *api='https://tagme.d4science.org/tagme/spot'*)

Find possible mentions in a text, do not link them to any entity.

Parameters

- **text** – the text where to find mentions.
- **gcube_token** – the authentication token provided by the D4Science infrastructure.
- **lang** – the Wikipedia language.
- **api** – the API endpoint.

hate_tweet_map.tweets_processor.MyTagMe.**normalize_title**(*title*)

Normalize a title to Wikipedia format. E.g. "barack Obama" becomes "Barack_Obama"

Parameters **title** – a title to normalize.

hate_tweet_map.tweets_processor.MyTagMe.**relatedness_title**(*tt_pairs*, *gcube_token=None*, *lang='en'*, *api='https://tagme.d4science.org/tagme/rel'*)

Get the semantic relatedness among pairs of entities. Entities are indicated by their Wikipedia ID (an integer).

Parameters

- **tt_pairs** – either one pair or a list of pairs of entity titles.
- **gcube_token** – the authentication token provided by the D4Science infrastructure.
- **lang** – the Wikipedia language.
- **api** – the API endpoint.

hate_tweet_map.tweets_processor.MyTagMe.**relatedness_wid**(*wid_pairs*, *gcube_token=None*, *lang='en'*, *api='https://tagme.d4science.org/tagme/rel'*)

Get the semantic relatedness among pairs of entities. Entities are indicated by their Wikipedia ID (an integer).

Parameters

- **wid_pairs** – either one pair or a list of pairs of Wikipedia IDs.
- **gcube_token** – the authentication token provided by the D4Science infrastructure.
- **lang** – the Wikipedia language.
- **api** – the API endpoint.

hate_tweet_map.tweets_processor.MyTagMe.**title_to_uri**(*entity_title*, *lang='en'*)
Get the URI of the page describing a Wikipedia entity.

Parameters

- **entity_title** – an entity title.
- **lang** – the Wikipedia language.

1.1.2 Entity Linker Service

Entity Linker

class hate_tweet_map.tweets_processor.EntityLinker.**EntityLinker**(*path_to_cnfg_file*)

tag(*raw_tweet: str*, *lang: str*) → list
This method send the text to tag on TagMe service and return the response :param raw_tweet: the text of the tweet to tag :type raw_tweet: str :param lang: tha language to use to tag the entities :type lang: str :return: a list with the id of the entities, the title and the url of the wikipedia page. :rtype: list

1.1.3 Tweet Processor

class hate_tweet_map.tweets_processor.TweetProcessor.**ProcessTweet**(*path_to_cnfg_file*)

__feel_it_analyze_sentiment(*tweet_text: str*, *tweet: dict*) → Tuple[int, dict, dict]
This method use the feel-it algorithms to perform the sentiment and emotion analysis.

NB. these models works only with italian language.

Parameters

- **tweet_text** (*str*) – the text of the tweet
- **tweet** (*dict*) – a dict representing the tweet

Returns the id of the process: 3; a dictionary that contains the result of the analysis; the tweet analyzed

Return type Tuple[int, dict, dict]

__get_osm_coordinates(*tweet: dict*, *user_location: Optional[str]*, *city: Optional[str]*, *country: Optional[str]*) → Tuple[int, bool, dict, dict]
This method use the Open Street Map service to find the coordinates of a specific location.

Parameters

- **tweet** (*dict*) – the tweet analyzed
- **user_location** (*str, optional*) – the user_location if the tweet contains it
- **city** (*str, optional*) – the city from where the tweet has been posted if the tweet contains it
- **country** (*str, optional*) – the country from where the tweet has been posted if the tweet contains it

Returns the id of the process: 5; a dictionary that contains the result of the analysis; the tweet analyzed

Return type Tuple[int, bool, dict, dict]

__link_entity(*tweet_text: str*, *tweet: dict*) → Tuple[int, dict, dict]
This method use the sent-it uniba service to perform the sentiment analyses of the tweet.

Parameters

- **tweet_text** (*str*) – the text of the tweet
- **tweet** (*dict*) – a dictionary representing the tweet

Returns the id of the process:1; a dictionary representing the result of the analyses; a dictionary representing the original tweet.

Return type Tuple[int, dict, dict]

__process_text_with_spacy(*text_tweet: str*, *tweet: dict*) → Tuple[int, dict, dict]
This method perform the natural language processing on the tweet text using spacy.

Parameters

- **text_tweet** (*str*) – the text of the tweet
- **tweet** (*dict*) – a dict representing the tweet

Returns the id of the process: 4; a dictionary that contains the result of the analysis; the tweet analyzed

Return type Tuple[int, dict, dict]

__save(*fut: concurrent.futures._base.Future*)
This is the callback function. when a tweets finish to be processed in it's thread this function retrieve the result returned by the function called by the thread, unpack it, adds the information on the tweet and save it in the db.

Parameters **fut** (*Future*) – the future object that contains the transformation done on the tweet

Returns None

__sent_it_analyze_sentiment(*tweet_text: str*, *tweet: dict*) → Tuple[int, dict, dict]
This method use the sent-it uniba service to perform the sentiment analyses of the tweet:

Parameters

- **tweet_text** (*str*) – the text of the tweet
- **tweet** (*dict*) – a dictionary representing the tweet

Returns the id of the process:2; a dictionary represent the result of the analysis (empty if something goes wrong); a dictionary representing the original tweet.

Return type Tuple[int, dict, dict]

__sent_it_analyze_sentiment2(*tweets: list*) → Tuple[int, list, list]
This method use the sent-it uniba service to perform the sentiment analyses of the tweet:

Parameters **tweets** (*list*) – list of tweets to send to sent_it

Returns the id of the process:2; a list represent the result of the analysis (empty if something goes wrong); a list representing the original tweets.

Return type Tuple[int, list, list]

start()
This method start the process on the tweets in according with the configuyration. For easch phase enabled in the config file the method retrieve the tweets to analyze, do the analyses and save all tweets process on the database. So after each phase the tweets processed are updated in the db. When the field all_tweets is set to False in the config file for each phase are retrieved from the database the tweets were the processed field is False and that have not already been processed by that phase (eg. for sentiment analyses that have not the field sentiment or for geo that have not the field geo.coordinates). Instead if the value of all_tweets is True all the tweets in the db are processed. :return: None

1.2 Tweets Searcher

1.2.1 Search Tweets

class hate_tweet_map.tweets_searcher.SearchTweets.**SearchTweets**(*mongodb: hate_tweet_map.database.DataBase*, *path_to_cnfg_file: str*)

__build_query(*user: Optional[str] = None*) → None
This method build the query to send to twitter

Parameters **user** (*str, optional*) – the id or name of the user whose tweets you want, defaults to None

Returns None

__connect_to_endpoint(*retried: bool = False*) → dict
This method sends the request to twitter and return the response. The possibles status codes in the twitter response are:

- 200: ok,in this case the response is a valid response;
- 429: rate limit exceeded, this means that either more requests were sent per second than allowed or more requests were sent in 15min than allowed. so in this case this method waits 1 second and tries to send the request again, if twitter still replies with a 429 code, it retrieves from the reply the time when the limit will reset and wait for that time to resubmit the request;
- 503: service overloaded, this means that twitter can't response to our requesst because there too many request to process. In this case this method wait for a minute and then retry to send the request.
- others: in this case the method raises an exception

Parameters **retried** (*bool, optional*) – a parameter that indicate if it is the first retry after an error or not, defaults to False

Raises **Exception** – when twitter response with not 200 or 429 status code.

Returns dict that contains the response from twitter

Return type dict

__make() → None
This method sends the request to twitter, elaborates it and saves the response. After the first search the number of tweets contained in the response are checked, if this number is equal to the number of result wanted set in the config file the method stop to send request. If this number is less than the number of result wanted set in the config file, the difference between the two number are done and a new request with this number as max_result query field are send, so this method a called with result_obtained_yet parameter

updated. Note that if the difference between the number of tweets obtained and the number of tweets wanted is greater than 500 the max_result query field for the next request is set to 500 instead if is less than 10 the max_result query field for the next request is set to 10. Moreover if the all_tweets parameters is set to True on the file config this method resend the request to twitter asking for 500 tweets per time (max_result = 500) until the end of the result is not reached.

Returns None

__next_page(*next_token=''*) → None
Insert in the query the token to obtain the next page of the tesult of the search.

Parameters **next_token** (*str, optional*) – the token obtained from twitter to reach the next page of the search

Returns None

__save()
THis method are called after that a request have been sent to twitter. When called this method process all the tweets received in parallel using the multithreading and then save all tweets processed on the database. Note that process only the tweet not already in the database.

Returns None

search() → int
This method start the search on twitter. So first build the query and then send it to twitter. If are set in the config file more users for each user tries to retrieve the number of tweets set in n_result config file field, only after reach this number perform the search on the next user.

Returns the number of the total tweets saved

Return type int

1.3 Users Searcher

1.3.1 Search Users

class hate_tweet_map.users_searcher.SearchUsers.**UserSearch**(*path_to_cnfg_file*)

__build_query(*users: str*) → None
This method build the query to send to the twitter api.

Parameters **users** (*str*) – a string that contains all the user's ids to search separated by a comma

Returns None

__connect_to_endpoint(*retried=False*) → dict
This method sends the request to twitter and return the response. The possibles status codes in the twitter response are:

- 200: ok,in this case the response is a valid response;
- 429: rate limit exceeded, this means that either more requests were sent per second than allowed or more requests were sent in 15min than allowed. so in this case this method waits 1 second and tries to send the request again, if twitter still replies with a 429 code, it retrieves from the reply the time when the limit will reset and wait for that time to resubmit the request;
- others: in this case the method raises an exception

Parameters **retried** (*bool, optional*) – a parameter that indicate if it is the first retry after an error or not, defaults to False

Raises **Exception** – when twitter response with not 200 or 429 status code.

Returns dict that contains the response from twitter

Return type dict

__make() → None
This method calls the method that send the request to twitter than elaborates and save the response.

Returns None

__retrieve_users_id() → None
This method retrieve all the id of the tweets from the database, than retrieve the id of the users already saved, subtract from the users obtained from the tweets the users already saved and finally save these user's ids in a list

__save() → None
This method process in parallel all the tweets received from one request and save all processed tweets in the db.

:return None

search() → None
This method search, elaborate and save the users on the database. The Twitter Search Users endpoint allow to search for 100 users per request. If the users to search are more than 100 the number of users to search are split in subset of 100 and the relative requests are sent.

Returns None

1.4 Manage Tweets

1.4.1 Manage Tweets

script.manage_tweets.manage_tweets.**main**()

Using this script is possible:

- extract some tweets from the database and save it on .json or .csv file
- delete some tweets

The criteria to select the tweets to extract/delete are defined in the manage_tweets.config file. Is possible modify that file to set the criteria. The possible criteria are:

- contains some specific word/words. In this case it is possible or write a list of words separated by comma in the words field, or use a txt file and write it path in the path field.
- contains a specific sentiment
- contains a word with a specific Part Of Speech (POS)
- raw criteria: a query written in mongodb style

These criteria and the words specified in the relative field/file are connected with the "OR" logical operator or with the "AND" logical operator. It is possible specify which operator must be used setting the logical_operator field in the config file.

Returns None

CHAPTER

TWO

USER GUIDE

2.1 Initialization

2.1.1 Setup

To use the Hate Tweet Map tools it's sufficient install the requires module, so just open a terminal, move into the root directory of the project and run:

```
python setup.py install
python setup.py install_lib
```

That's all.

2.2 Scripts

2.2.1 Search Tweets Script

2.2.1.1 Configuration file

To search tweets on twitter the first thing to do is edit the configuration file search_tweets.config in the script/search_tweets folder. The configuration file looks like this:

```
mongodb:
    # default url
    url: mongodb://localhost:27017/
    database:
    collection:
twitter:
    configuration:
        barer_token: AAAAAAAAAAAAAAAAAAAAAAPtPgEAAAAAoVlZ4I0szkcu4dL%2Bhqif%2F%2BF45Oo
↪%3DJbvSo773bskLu1GexDv9Dq1HjuSjfSwfxgLdDXEdlPO5mKyE6G
        end_point: https://api.twitter.com/2/tweets/search/all
    search:
        # MANDATORY:
        # Please fill at least one of the following fields. If both fields are set it's␣
↪possible to search for a twitter with the given keyword tweeted by the specific user.

        # enter the keyword/s to search for on twitter. It's also possible use logical␣
↪operators. If no logical operator are specified all keywords will be searched in AND.
```

(continues on next page)

(continued from previous page)

```
        # the AND operator is handle by a space, so to search "Joe AND Trump" just write
↪"Joe Trump", the OR operator is "OR".
        # for example: "Joe Biden", "Biden OR Trump", "(Biden OR Trump) whitehouse" (the␣
↪last query means: "(Biden OR Trump) AND whitehouse").
        keyword:

        # enter the username or the user id to search for tweets of a specific user.
        user:

        #OPTIONAL:
        # the language of the tweets
        lang:

        # enable/disable the twitter context annotation in the twitter response
        context_annotations: True

        # the max results of tweets
        n_results: 10
        # possible value: True/False
        # if this field is set to True the value on n_result it automatically overwrite␣
↪and set to 500.
        all_tweets: False
        # please see here for information about time fields: https://developer.twitter.
↪com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all
        # you can set:
            # 1. only start_time: if you specify only start time but no end time, end␣
↪time will be assumed to be current time (-30 sec). (see https://twittercommunity.com/t/
↪twitter-api-v2-search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
            # 2. only end_time: If you specify only end time, start time will be assumed␣
↪30 days before the end time specified. (see https://twittercommunity.com/t/twitter-api-
↪v2-search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
            # 3. both: the tweets in the range specified
            # 4. none: By default, a request will return Tweets from up to 30 days ago␣
↪if you do not include this parameters. (see https://developer.twitter.com/en/docs/
↪twitter-api/tweets/search/api-reference/get-tweets-search-all)
        time:
            # format: YYYY-MM-DDTHH:mm:ssZ (ISO 8601/RFC 3339)
            # example value: 2018-10-19T07:20:50.52+00:00
            start_time:
            end_time:
        # geo parameter.
        # only one of the following fields could be set
        geo:
            place:
            place_country:
            # example value: -105.301758 39.964069 -105.178505 40.09455
            bounding_box:
            # please if you want search by point radius set all the parameters in the␣
↪point_radius section.
            point_radius:
                # example value: 2.355128
                longitude:
```

(continues on next page)

(continued from previous page)

```
                # 48.861118
                latitude:
                # 16km
                radius:
        #Possible values: True/False. When is True only tweet that are not retweet are␣
↪retrieved. default value: False.
        filter_retweet: False
```

2.2.1.1.1 Mandatory Section

The mandatory section is this one:

```
search:
        # MANDATORY:
        # Please fill at least one of the following fields. If both fields are set it's␣
↪possible to search for a twitter with the given keyword tweeted by the specific user.

        # enter the keyword/s to search for on twitter. It's also possible use logical␣
↪operators. If no logical operator are specified all keywords will be searched in AND.
        # the AND operator is handle by a space, so to search "Joe AND Trump" just write
↪"Joe Trump", the OR operator is "OR".
        # for example: "Joe Biden", "Biden OR Trump", "(Biden OR Trump) whitehouse" (the␣
↪last query means: "(Biden OR Trump) AND whitehouse").
        keyword:

        # enter the username or the user id to search for tweets of a specific user.
        user:
```

As explained in the comments in the keywords section it is possible set the keyword (s) that tweets must contain. To search using the logical operator just use the parentheses and the keyword OR and the space for AND.

In the user field you can enter the user's ID or username. Note that at least one of these two fields must be set. It is also possible to set both fields, which means "search tweets containing this [keyword] from this [user]".

2.2.1.1.2 Optional Section

The optional section is:

```
#OPTIONAL:
    # the language of the tweets
    lang:

    # enable/disable the twitter context annotation in the twitter response
    context_annotations: True

    # the max results of tweets
    n_results: 10
    # possible value: True/False
    # if this field is set to True the value on n_result it automatically overwrite and␣
↪set to 500.
```

(continues on next page)

(continued from previous page)

```
    all_tweets: False
    # please see here for information about time fields: https://developer.twitter.com/
↪en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all
    # you can set:
        # 1. only start_time: if you specify only start time but no end time, end time␣
↪will be assumed to be current time (-30 sec). (see https://twittercommunity.com/t/
↪twitter-api-v2-search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
        # 2. only end_time: If you specify only end time, start time will be assumed 30␣
↪days before the end time specified. (see https://twittercommunity.com/t/twitter-api-v2-
↪search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
        # 3. both: the tweets in the range specified
        # 4. none: By default, a request will return Tweets from up to 30 days ago if␣
↪you do not include this parameters. (see https://developer.twitter.com/en/docs/twitter-
↪api/tweets/search/api-reference/get-tweets-search-all)
    time:
        # format: YYYY-MM-DDTHH:mm:ssZ (ISO 8601/RFC 3339)
        # example value: 2018-10-19T07:20:50.52+00:00
        start_time:
        end_time:
    # geo parameter.
    # only one of the following fields could be set
    geo:
        place:
        place_country:
        # example value: -105.301758 39.964069 -105.178505 40.09455
        bounding_box:
        # please if you want search by point radius set all the parameters in the point_
↪radius section.
        point_radius:
            # example value: 2.355128
            longitude:
            # 48.861118
            latitude:
            # 16km
            radius:
    #Possible values: True/False. When is True only tweet that are not retweet are␣
↪retrieved. default value: False.
    filter_retweet: False
```

As shown is composed by 7 sub-section:

1. *Lang*
2. *Context Annotation*
3. *Number of results*
4. *Reach all tweets*
5. *Time*
6. *Geo*
7. *Filter retweet*

2.2.1.1.2.1 1. Lang

```
# the language of the tweets
lang:
```

This field indicate the language of the tweets that you want retrieve.

From Twitter Api Doc:

Restricts tweets to the given language, given by an ISO 639-1 code . *Language detection is best-effort.*

An example values: it, en, pt, es.

Possible values: *any ISO-639-1 code*

2.2.1.1.2.2 2. Context Annotation

```
# enable/disable the twitter context annotation in the twitter response
context_annotations: True
```

This field indicate to Twitter to include or not the context annotation for tweet that have it.
If this field is set to `True` the `n_results` field will be se automatically yo 100 in according to the TwitterAPI doc.
For more information see the official doc here.

Possible values: *True/False*

2.2.1.1.2.3 3. Number of results

```
# the max results of tweets
n_results: 10
```

This field indicate to Twitter to how may tweets the response should contain.
Twiiter allow to search for minimum 10 tweets to maximum 500 tweets for request.
So if the value insert in this field is less than 10 this field automatically will be set to 10; if the value insert is greater than 500 more requests will be send to Twitter.
Note that Twitter to reach a number of tweets as close as possible to the value given here.

Possible values: *any int number*.

N.B

If the `all_tweets` field is set to True this field automatically will be set to 500 whatever value is insert here.
If the `context_annotation` field is set to True this field automatically will be set to 100 whatever value is insert here.
If the `all_tweets` field is set to True and `context_annotation` field is set to True this field automatically will be set to 100 whatever value is insert here.

2.2.1.1.2.4 4. Reach all tweets

```
# if this field is set to True the value on n_result it automatically overwrite and set␣
↪to 500.
all_tweets: False
```

When a research is send to Twitter it responds with the number of tweets asked and, if possible, with a `next_token`, this token allow to go to the next page of results.
So this field indicate to the script to iterate all over the pages returned by Twitter.
In this case the `n_results` field will be set automatically to 500 to obtain 500 tweets per time.

Possible values: *True/False*.

N.B
Setting this field to True means start a very time expensive research.

2.2.1.1.2.5 5. Time

```
# please see here for information about time fields: https://developer.twitter.com/en/
↪docs/twitter-api/tweets/search/api-reference/get-tweets-search-all
# you can set:
   # 1. only start_time: if you specify only start time but no end time, end time will␣
↪be assumed to be current time (-30 sec). (see https://twittercommunity.com/t/twitter-
↪api-v2-search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
   # 2. only end_time: If you specify only end time, start time will be assumed 30 days␣
↪before the end time specified. (see https://twittercommunity.com/t/twitter-api-v2-
↪search-endpoint-what-is-start-time-and-end-time-actual-default/152679)
   # 3. both: the tweets in the range specified
   # 4. none: By default, a request will return Tweets from up to 30 days ago if you do␣
↪not include this parameters. (see https://developer.twitter.com/en/docs/twitter-api/
↪tweets/search/api-reference/get-tweets-search-all)
time:
   # format: YYYY-MM-DDTHH:mm:ssZ (ISO 8601/RFC 3339)
   # example value: 2018-10-19T07:20:50.52+00:00
   start_time:
   end_time:
```

This field allow to search tweets in a specific range of time.
There 4 possible configuration:

1. only `start_time`: if you specify only start time but no `end_time`, `end_time` will be assumed to be current time (-30 sec).
2. only `end_time`: if you specify only `end_time`, `start_time` will be assumed 30 days before the `end_time` specified.
3. both: the tweets in the range specified
4. none: by default, a request will return tweets from up to 30 days ago if you do not include this parameters.

For more information see:

- https://twittercommunity.com/t/twitter-api-v2-search-endpoint-what-is-start-time-and-end-time-actual-default/152679
- https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all

The values in this fields must be in the ISO 8601/RFC 3339 format, so: *YYYY-MM-DDTHH:mm:ss+Z.*
An example value is: *2018-10-19T07:20:50.52+00:00* where *00:00* is the time zone.

Possible values: *any date in ISO 8601/RFC 3339 format.*

2.2.1.1.2.6 6. Geo

```
geo:
      place:
      place_country:
      # example value: -105.301758 39.964069 -105.178505 40.09455
      bounding_box:
      # please if you want search by point radius set all the parameters in the point_
↪radius section.
      point_radius:
          # example value: 2.355128
          longitude:
          # 48.861118
          latitude:
          # 16km
          radius:
```

In this section it is possible to set the geographical parameters, in this way it is possible to filter the tweets based on their geographical origin.
The possible parameters are, please note that **Only one of these fields must be set**:

- *place*:
matches tweets tagged with the specified location or twitter place ID. Multi-word place names (“New York City”, “Palo Alto”) should be enclosed in quotes.

Possible values: *any name of city, enclosed in quotes if the place names ia s multi/word.*

- *place_country*:
attaches tweets where the country code associated with a tagged place/location matches the given ISO alpha-2 character code.
You can find a list of valid ISO codes here

Possible values: *any name of country in ISO_3166-1_alpha-2 format.*

- *bounding_box*:
matches against the place.geo.coordinates object of the Tweet when present, and in Twitter, against a place geo polygon, where the place polygon is fully contained within the defined region.
`bounding_box: west_long south_lat east_long north_lat`

- `west_long south_lat` represent the southwest corner of the bounding box where `west_long` is the longitude of that point, and `south_lat` is the latitude.
- `east_long north_lat` represent the northeast corner of the bounding box, where `east_long` is the longitude of that point, and `north_lat` is the latitude.
- Width and height of the bounding box must be less than 25mi
- Longitude is in the range of ± 180
- Latitude is in the range of ± 90
- All coordinates are in decimal degrees.

Possible values: *4 coordinates in decimal degrees.*
Example: `bounding_box: -105.301758 39.964069 -105.178505 40.09455`

- *point_radius*:
matches against the place.geo.coordinates object of the Tweet when present, and in Twitter, against a place geo polygon, where the Place polygon is fully contained within the defined region.

- **longitude:**

  longitude is in the range of ± 180
  Possible values: *a coordinate in decimal degrees.*
  Example: `longitude: 48.861118`

- **latitude:**

  latitude is in the range of ± 90
  Possible values: *a coordinate in decimal degrees.*
  Example: `longitude: 48.861118`

- **radius:**

  radius must be less than 25mi; units of radius supported are miles (mi) and kilometers (km); radius must be less than 25mi

Possible values: *an integer number followed by the string 'km' or 'mi' to indicate if the value refer to kilometers or miles.*

Example: `radius: 10km`

See here for more information:

- https://developer.twitter.com/en/docs/tutorials/filtering-tweets-by-location
- https://developer.twitter.com/en/docs/twitter-api/tweets/search/integrate/build-a-query

2.2.1.1.2.7 7. Filter retweet

```
#Possible values: True/False. When is True only tweet that are not retweet are retrieved.
↪ default value: False.
filter_retweet: False
```

This field indicate to Twitter to include or not the retweet in the response.
If is True Twitter response could contain also retweet, if false not.

Possible values: *True/False*

2.2.1.2 Use the script

After editing and setting the configuration file just open a terminal in the folder script/search_tweets and launch this command:

```
python search_tweets.py
```

2.2.2 Search Users Script

This script allow to search information about a list of users. Specfically this tools read the tweets from a collection, save the users id and then search the information about these users.

2.2.2.1 Configuration file

To search users on twitter the first thing to do is edit the configuration file search_users.config in the script/search_users folder. The configuration file looks like this:

```
mongodb_tweets:
    url: mongodb://localhost:27017/
    database:
    collection:
mongodb_users:
    url: mongodb://localhost:27017/
    database:
    collection:

twitter:
```

(continues on next page)

(continued from previous page)

```
    configuration:
        barer_token: AAAAAAAAAAAAAAAAAAAAAAPtPgEAAAAAoVlZ4I0szkcu4dL%2Bhqif%2F%2BF45Oo
↪%3DJbvSo773bskLu1GexDv9Dq1HjuSjfSwfxgLdDXEdlPO5mKyE6G
        end_point: https://api.twitter.com/2/users
```

2.2.2.1.1 Mongodb tweets

```
mongodb_tweets:
    url: mongodb://localhost:27017/
    database:
    collection:
```

This section contains information necessary to connect to the mongo db collection where the tweets are saved and retrieve from it the users ID.

2.2.2.1.2 Mongodb Users

```
mongodb_users:
    url: mongodb://localhost:27017/
    database:
    collection:
```

This section contains information necessary to connect to the mongo db collection where save the users information obtained from Twitter.

2.2.2.1.3 Twitter

```
twitter:
    configuration:
        barer_token: AAAAAAAAAAAAAAAAAAAAAAPtPgEAAAAAoVlZ4I0szkcu4dL%2Bhqif%2F%2BF45Oo
↪%3DJbvSo773bskLu1GexDv9Dq1HjuSjfSwfxgLdDXEdlPO5mKyE6G
        end_point: https://api.twitter.com/2/users
```

This section contains information necessary to connect to Twitter API. Don't change the value of `end_point` field if you really don't know what are you doing. The `barer_token` field it's related to an Twitter App with research privileges.

2.2.2.2 Use the script

After editing and setting the configuration file just open a terminal in the folder script/search_users and launch this command:

```
python search_users.py
```

2.2.3 Process Tweets Script

This script allow to perform 5 different types of analysis on thw saved tweets. The possible analysis are:

- *Entity Linker*: uses the TagMe service to find entities in the text of the tweet and to connect these with the respective wikipedia page.
- *Geo*: if present uses the geographic information in the tweet to find the coordinates of the place where the tweet have benn posted. Uses Open Street Map service. (This operation could be time expensive cause OSM allows to send only one request per second.)
- *Natural Language Processing*: uses spacy model to lemmatize the text of the tweet. In addition save the POS and the Morphological information and the entities found by spacy in the text.
- *Sentiment Analysis*: uses two different services, sent-it and feel-it, to perform the sentiment analysis of the tweet. Note that feel/it algorithm works only with italian tweets.

Note that there are two mode to select the tweets to analyze:

- all the tweets in the collection
- only the tweets that have not yet been passed to the Natural Language Phase.

To choose the first mode just set the `analyze_all_tweets` to `True` otherwise to `False`.

To better understand this mechanism see this:

```
If analyze_all_tweets is False
------------------ If geocode=True
-----------------------------------look for tweet not geocoded yet
-----------------------------------geocode the tweet (regardless NLP tasks)
------------------ If NLP=True
--------------------------------- look for tweet not processed
--------------------------------- process the tweet
etc.

If analyze_all_tweets is True
------------------ If geocode=True
---------------------------------- geocode all the tweets (and overwrite previous␣
↪information)
------------------ If NLP=True
---------------------------------- process again all the tweets (and overwrite previous␣
↪information)
etc.

accordingly,
if I set: analyze_all_tweets=True, geocode=True, nlp=False, I only run geocode on all␣
↪the tweets, regardless the state of the NLP processing.
```

2.2.3.1 Configuration file

To process tweets the first thing to do is to edit the configuration file process_tweets.config in the script/tweets_processor folder. The configuration file looks like this:

```
mongodb:
    url: mongodb://localhost:27017/
    database:
    collection:
analyses:
    nlp: True
    tagme:
        enabled: True
        token: 7f5391f2-142e-4fd5-9cc9-56e91c4c9add-843339462
        lang: it
        is_twitter: True
        rho_value: 0.15
    sentiment_analyze:
        sent_it: True
        feel_it: True
    geocoding: True
    analyze_all_tweets: False
```

2.2.3.1.1 Mongodb

```
mongodb:
    url: mongodb://localhost:27017/
    database:
    collection:
```

This section provide the information to connect to the mongodb collection where the tweets to process are saved.

2.2.3.1.2 Analyses:Nlp

```
analyzes:
   nlp: True
```

This section enables or disables the SpaCy's Natural Language Processing.
Possible values: True/False

2.2.3.1.3 Analyses:TagMe

```
analyses:
   tagme:
       enabled: True
       token: 7f5391f2-142e-4fd5-9cc9-56e91c4c9add-843339462
       is_tweet: True
       rho_value: 0.15
```

This section enables Entity Linker phase using TagMe service.

- **enabled::**

  enable or disable this phase.
  Possible values: True/False

- **token::**

  the token obtained from TagMe to send the requests. See here for more info.
  Possible values: a valid TagMe token

- **is_tweet::**

  indicate to TagMe service if the text given is a tweet or not.
  Possible values: True/False

- **rho_value::**

  estimates the confidence in the annotation. (Note that does not indicate the relevance of the entity in the input text). You can use the value to discard annotations that are below a given threshold. The threshold should be chosen in the interval [0,1]. A reasonable threshold is between 0.1 and 0.3.
  Possible values: any number between 0 and 1

2.2.3.1.4 Analyses:Sentiment Analyses

```
analyses:
   sentiment_analyze:
          sent_it: True
          feel_it: True
```

This section enables Sentiment Analyses phase.

- **sent-it::**

  enable or disable sent-it phase.
  Possible values: True/False

- **feel-it::**

  enable or disable sent-it phase.
  Note that this phase will disable automatically in presence of english tweet.
  Possible values: True/False

2.2.3.1.5 Analyses:Geocoding

```
geocoding: True
```

This section enables or disables the geocoding phase using Open Street Map service.
Possible values: True/False

2.2.3.1.6 Analyses:Analyze all tweets

```
analyze_all_tweets: False
```

This section indicate to analyze all tweets in the mongodb collection or not.
Possible values: True/False

2.2.3.2 Use the script

After editing and setting the configuration file just open a terminal in the folder script/process_tweets and launch this command:

```
python process_tweets.py
```

2.2.4 Manage Tweets Script

Using this script is possible:

- extract some tweets from the database and save it on .json or .csv file
- delete some tweets

The criteria to select the tweets to extract/delete are defined in the manage_tweets.config file. Is possible modify that file to set the criteria. The possible criteria are:

- contains some specific word/words. In this case it is possible or write a list of words separated by comma in the words field, or use a txt file and write it path in the path field.
- contains a specific sentiment
- contains a word with a specific Part Of Speech (POS)
- raw criteria: a query written in mongodb style

These criteria and the words specified in the relative field/file are connected with the "OR" logical operator or with the "AND" logical operator. It is possible specify which operator must be used setting the logical_operator field in the config file.

2.2.4.1 Configuration file

To process tweets the first thing to do is to edit the configuration file process_tweets.config in the script/manage_tweets folder. The configuration file looks like this:

```
mongodb:
    url: mongodb://localhost:27017/
    database:
    collection:
#possible values: extract delete
mode:
#json or csv (only with extract mode)
format:
criteria:
  #possible values: negative positive neutral
  sentiment:
  #a list of keywords separated by a comma
  keywords:
    words:
    path:
  postag:
  #a raw NoSql query
  raw_query:
  #possible value: and or. This field specify with which logical operator the fields␣
↪must be connected
  logical_operator: or
```

2.2.4.1.1 Mongodb

```
mongodb:
    url: mongodb://localhost:27017/
    database:
    collection:
```

This section provide the information to connect to the mongodb collection where the tweets to manage are saved.
Mandatory

2.2.4.1.2 Mode

```
#possible values: extract delete
   mode: extract
```

The mode indicates what the script have to do. As explain before it's possible extract and save in a file the tweets or delete it.

Possible values: extract delete

Mandatory

2.2.4.1.2.1 Mode: extract

```
#possible values: extract delete
  mode: extract
  #json or csv (only with extract mode)
  format:
```

To extract tweets it's necessary set the `mode: extract` and choose a format so: or `format: csv` or `format: json`.
This will generate a file (in the format chosen) in the working directory.

2.2.4.1.2.2 Mode: delete

```
#possible values: extract delete
  mode: delete
  #json or csv (only with extract mode)
  format:
```

To delete tweets it's necessary set the `mode: delete` and leaves blank the `format` field.

2.2.4.1.3 Criteria

```
criteria:
  #possible values: negative positive neutral
  sentiment:
  #a list of keywords separated by a comma
  keywords:
    words:
    path:
  postag:
  #a raw NoSql query
  raw_query:
  #possible value: and or. This field specify with which logical operator the fields␣
↪must be connected
  logical_operator: or
```

This section set the criteria to find the tweets in the db (to delete or extract it)

- **`sentiment::`**

  setting this field it's possible retrieve tweets with a specific sentiment, in particular choosing between tweets with neutral or positive or negative sentiment.

  Possible values: negative/positive/neutral

  Optional

- **`keywords::`**

  setting this field it's possible retrieve tweets that contains specific words

N.B if the `logical_operator` it's set to `or` will be retrieved tweets that have one of the words spcified here,
otherwise if the field it's set to `and` will be retrieved only tweets that contains all the specified words.

- **`words::`**

  a list of words to search separeted by a comma
  Possible values: a list of words separated by a comma
  Example value: sun,sea,island
  Optional

- **`path::`**

  the path to a .txt file contained the words to search.
  The .txt file have to contain each word to search in a different line, example:

  ```
  sun
  sea
  island
  ```

  Possible values: a valid path to a .txt file
  Optional

- **`postag::`**

  setting this field it's possible retrieve tweets that contains a word with a specific POS tag.
  For more info see:

  - here for a generic understanding
  - here for a complete list of italian SpaCy's POS values (see labels scheme section in it_core_news_lg)
  - here for a complete list of english SpaCy's POS values (see labels scheme section in en_core_web_lg)

  Possible values: any valid POS value
  Example value: ADV
  Optional

- **`raw_query::`**

  setting this field it's possible to write a own query.
  the query must be a mongodb query and have to take in account the fields of the tweet saved in the collection.
  Possible values: any valid mongodb query
  Example value: {'processed':True}
  Optional

- **`logical_operator::`**

  if more than one criteria field are set or if the keywprds field it' set it's necessary to define how logically connect this criteria or the words specified, so if using `and` logical operator or `or` logical operator.
  Possible values: or/and
  Mandatory

2.2.4.2 Use the script

After editing and setting the configuration file just open a terminal in the folder script/manage_tweets and launch this command:

```
python manage_tweets.py
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

hate_tweet_map.tweets_processor.EntityLinker, 3
hate_tweet_map.tweets_processor.MyTagMe, 1
hate_tweet_map.tweets_processor.TweetProcessor, 3
hate_tweet_map.tweets_searcher.SearchTweets, 5
hate_tweet_map.users_searcher.SearchUsers, 6

s

script.manage_tweets.manage_tweets, 7

INDEX

Symbols

__build_query() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 5
__build_query() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 6
__connect_to_endpoint() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 5
__connect_to_endpoint() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 6
__feel_it_analyze_sentiment() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 3
__get_osm_coordinates() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 3
__link_entity() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4
__make() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 5
__make() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 7
__next_page() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 6
__process_text_with_spacy() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4
__retrieve_users_id() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 7
__save() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4
__save() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 6
__save() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 7
__sent_it_analyze_sentiment() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4
__sent_it_analyze_sentiment2() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4

A

annotate() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 1
AnnotateResponse (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1
Annotation (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1
as_pair() (*hate_tweet_map.tweets_processor.MyTagMe.Relatedness method*), 1

E

EntityLinker (*class in hate_tweet_map.tweets_processor.EntityLinker*), 3

G

get_annotations() (*hate_tweet_map.tweets_processor.MyTagMe.Annota method*), 1
get_mentions() (*hate_tweet_map.tweets_processor.MyTagMe.MentionsR method*), 1
get_relatedness() (*hate_tweet_map.tweets_processor.MyTagMe.Relate method*), 1

H

hate_tweet_map.tweets_processor.EntityLinker module, 3
hate_tweet_map.tweets_processor.MyTagMe module, 1
hate_tweet_map.tweets_processor.TweetProcessor module, 3
hate_tweet_map.tweets_searcher.SearchTweets module, 5
hate_tweet_map.users_searcher.SearchUsers module, 6

M

main() (*in module script.manage_tweets.manage_tweets*), 7

Mention (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1
mentions() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 2
MentionsResponse (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1
module
    hate_tweet_map.tweets_processor.EntityLinker, 3
    hate_tweet_map.tweets_processor.MyTagMe, 1
    hate_tweet_map.tweets_processor.TweetProcessor, 3
    hate_tweet_map.tweets_searcher.SearchTweets, 5
    hate_tweet_map.users_searcher.SearchUsers, 6
    script.manage_tweets.manage_tweets, 7

N

normalize_title() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 2

P

ProcessTweet (*class in hate_tweet_map.tweets_processor.TweetProcessor*), 3

R

Relatedness (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1
relatedness_title() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 2
relatedness_wid() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 2
RelatednessResponse (*class in hate_tweet_map.tweets_processor.MyTagMe*), 1

S

script.manage_tweets.manage_tweets
    module, 7
search() (*hate_tweet_map.tweets_searcher.SearchTweets.SearchTweets method*), 6
search() (*hate_tweet_map.users_searcher.SearchUsers.UserSearch method*), 7
SearchTweets (*class in hate_tweet_map.tweets_searcher.SearchTweets*), 5
start() (*hate_tweet_map.tweets_processor.TweetProcessor.ProcessTweet method*), 4

T

tag() (*hate_tweet_map.tweets_processor.EntityLinker.EntityLinker method*), 3
title_to_uri() (*in module hate_tweet_map.tweets_processor.MyTagMe*), 2

U

uri() (*hate_tweet_map.tweets_processor.MyTagMe.Annotation method*), 1
UserSearch (*class in hate_tweet_map.users_searcher.SearchUsers*), 6