Hatespeech detection

Generated by Doxygen 1.8.20

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 ai Namespace Reference

Functions

- def analyse_text (text, modelname="logistic_regression")
- def analyse_ad_lstm (modelname)
- def analyse_ad ()
- def process_text (text)
- def return_token (text)
- def construct_model (data, hate, modelname="logistic_regression")
- def logistic (vectorizer, data, hate, modelname)
- def parallel_construct (data, func)
- def naive_bayes (vectorizer, data, hate, modelname)
- def construct_lstm (data, hate, tokenizer, modelname, maxlen=500)
- def construct_lstm_tibo (data, hate, modelname)
- def construct_lstm_les (data, hate, tokenizer, modelname, maxlen=500)
- def make_lstm_model (x)
- def make_lstm_les_model (x)

5.1.1 Detailed Description

```
@package AI
The Ai module implements the core ai functions
```

5.1.2 Function Documentation

5.1.2.1 analyse_ad()

```
def ai.analyse_ad ( )

 a function to evaluate a new data set and the adversarials dataset with all calssic models

confusion matrix can be made with plot_confusion_matrix
```

5.1.2.2 analyse_ad_lstm()

```
def ai.analyse_ad_lstm (
              modelname )
```

a function to evaluate a new data set and the adversarials dataset with all lstm

to generate confusion matrix see plot_confusion_matrix

5.1.2.3 analyse_text()

```
def ai.analyse_text (
              text,
              modelname = "logistic_regression" )
```

a function to predict a tweet with logistic

used for the webinterface

5.1.2.4 construct_lstm()

```
def ai.construct_lstm (
              data,
              hate,
              tokenizer,
              modelname,
              maxlen = 500 )
```

make a lstm model

based on a online text classification example

5.1.2.5 construct_lstm_les()

```
def ai.construct_lstm_les (
              data,
              hate,
              tokenizer,
              modelname,
              maxlen = 500 )
```

make a lstm model and train it, save it

based on the excersise of the lesseon

5.1.2.6 construct_lstm_tibo()

```
def ai.construct_lstm_tibo (
              data,
              hate,
              modelname )
```

make a lstm model

based on https://www.youtube.com/watch?v=j7EB7yeySDw

5.1.2.7 construct_model()

```
def ai.construct_model (
              data,
              hate,
              modelname = "logistic_regression" )
```

a function to choose the model to build when a web requets comes in

for flask server

5.1.2.8 logistic()

```
def ai.logistic (
              vectorizer,
              data,
              hate,
              modelname )
```

make a logistic model

basedon original repo

5.1.2.9 make_lstm_les_model()

```
def ai.make_lstm_les_model (
              x )
```

make a lstm model

based on the excersise of the lesseon

5.1.2.10 make_lstm_model()

```
def ai.make_lstm_model (
              x )
```

make a lstm model

based on the an online lstm text classification example

5.1.2.11 naive_bayes()

```
def ai.naive_bayes (
              vectorizer,
              data,
              hate,
              modelname )
```

make a naive bayes model

based on the sklearn docs

5.1.2.12 parallel_construct()

```
def ai.parallel_construct (
              data,
              func )
```

a function to transform sequences in parallel

based on https://github.com/rafaelvalero/ParallelTextProcessing

5.1.2.13 process_text()

```
def ai.process_text (
              text )
```

return just the text preprocessing

For debugging

5.1.2.14 return_token()

```
def ai.return_token (
              text )
```

5.2 db Namespace Reference

Classes

- class DB

5.2.1 Detailed Description

```
@package DB
This module manages access to the sqllite databases, these provide better performance than CSV files.

More details.
```

5.3 NLP Namespace Reference

Classes

- class CustomTweetTokenizer

Functions

- def lemmatize (token, pos)
- def wordsegment (token)
- def spell_checker (token)
- def text_precessing (text)
- def basic_precessing (text)
- def basic_precessing_char (text)
- def text_precessing_char (text)
- def remove_repeats (word)
- def get_wordnet_pos (treebank_tag)
- def char_boundary (tokens)
- def has_word (word, voca)
- def lemmanize_text (tokens)

Variables

- reg = re.compile(r'(.)\1{2,}')
- mention_hashtag_regex = re.compile(r'([^\w]|^)@[\w\-]+|^[\s]#([^\s])+')
- url_remove
- contarction_not = re.compile(r'n\'t')
- contarction_am = re.compile(r'\'m')
- contarction_have = re.compile(r'\'ve')
- contarction_will = re.compile(r'\'ll')
- a = re.compile(r'(?:\b(?:@|/-\\|\^|/\\))')
- b = re.compile(r'(?:\b(?:\|:|P>|ß))')
- c = re.compile(r'(?:\b[©¢<\[({])')
- d = re.compile(r'(?:\b(?:\)|\|\)|\[\)|\?|\|>\|o))')
- e = re.compile(r'(?:\b(?:&|€|\[-))')
- f = re.compile(r'(?:\b(?:\|=|/=|ph|ƒ|\|#))')
- g = re.compile(r'(?:\b(?:&|C-|\(_\+))')
- h = re.compile(r'(?:\b(?:#|}{|\|-\||\]-\[\|\[-\]|\(-\)|\)-\(|/-/))')
- i = re.compile(r'(?:\b(?:!|\||\]|¡))')
- j = re.compile(r'(?:\b(?:¿|_\||_/|</|\(/))')
- k = re.compile(r'(?:\b(?:\|<|\|\{|\|\()')
- letter_l = re.compile(r'(?:\b(?:\||£|\|_|¬))')
- m = re.compile(r'(?:\b(?:\|v\||\|\\/\||/\\/\\|\(v\)|/\|\\|//\.|\^\^))')
- n = re.compile(r'(?:\b(?:\|\\\||/\\/|\[\\\]|<\\>|/V|\^/))')
- o = re.compile(r'(?:\b(?:\(\)|\[\]|°))')
- p = re.compile(r'(?:\b(?:\|*|¶|\|o|\|°|\|\"|\|>|\|\^\(o\)|\|\^\(\)))')
- q = re.compile(r'(?:\b(?:\(\)_|\(_,\)|<\|))')
- r = re.compile(r'(?:\b(?:\|\^|\|\?|®))')
- s = re.compile(r'(?:\b(?:\$|§))')
- t = re.compile(r'(?:\b(?:\+|-\|-|†|\'\[\]\'))')
- u = re.compile(r'(?:\b(?:\|_\||\(_\)))')
- v = re.compile(r'(?:\b(?:\\/|\^))')
- w = re.compile(r'(?:\b(?:VV|\\/\\/|\\\\\'|\'//|\\\|/|\\\^/))')
- x = re.compile(r'(?:\b(?:><|\)\(|%))')
- y = re.compile(r'(?:\b(?:¥|\'/))')
- z = re.compile(r'(?:\b(?:~/_|-/_|>_))')
- checker = SpellChecker()
- wnl = WordNetLemmatizer()
- tag = nltk.pos_tag
- stopwords_set = stopwords.words("english")
- known_words = set(words.words())
- database = DB()
- dictionary hate = {i[0] for i in database.db_load_lexicon()}
- maxsize
- tokenize = CustomTweetTokenizer().tokenize

5.3.1 Detailed Description

```
@package NLP
Natural language processing

Does all pre proccesing
```

5.3.2 Function Documentation

5.3.2.1 basic_precessing()

```
def NLP.basic_precessing (
              text )
```

basic pre processing

for comparing results

5.3.2.2 basic_precessing_char()

```
def NLP.basic_precessing_char (
              text )
```

basic pre processing

for comparing results

5.3.2.3 char_boundary()

```
def NLP.char_boundary (
              tokens )
```

revert char boundary attack

uses lexicon to check for hate in a unknown word

5.3.2.4 get_wordnet_pos()

```
def NLP.get_wordnet_pos (
              treebank_tag )
```

convert pos tags to lemmanitizer wordnet pos tags

helper function of lemmanize_text()

5.3.2.5 has_word()

```
def NLP.has_word (
              word,
              voca )
```

uses lexicon to check for hate word in a unknown word

helper function of char_boundary() function

5.3.2.6 lemmanize_text()

```
def NLP.lemmanize_text (
              tokens )
```

lemmanitize tokens

use of pos tagger to detect nouns, verbs, ...

5.3.2.7 lemmatize()

```
def NLP.lemmatize (
              token,
              pos )
```

5.3.2.8 remove_repeats()

```
def NLP.remove_repeats (
              word )
```

remove repeats

5.3.2.9 spell_checker()

```
def NLP.spell_checker (
              token )
```

5.3.2.10 text_precessing()

```
def NLP.text_precessing (
              text )
```

iMain entry for test preprocessing

5.3.2.11 text_precessing_char()

```
def NLP.text_precessing_char (
              text )
```

for use in char-based models

5.3.2.12 wordsegment()

```
def NLP.wordsegment (
              token )
```

5.3.3 Variable Documentation

5.3.3.1 a

```
NLP.a = re.compile(r'(?:\b(?:@|/-\\|\^|/\\))')
```

5.3.3.2 b

```
NLP.b = re.compile(r'(?:\b(?:\|:|P>|ß))')
```

5.3.3.3 c

```
NLP.c = re.compile(r'(?:\b[©¢<\[({])')
```

5.3.3.4 checker

```
NLP.checker = SpellChecker()
```

5.3.3.5 contarction_am

```
NLP.contarction_am = re.compile(r'\'m')
```

5.3.3.6 contarction_have

```
NLP.contarction_have = re.compile(r'\'ve')
```

5.3.3.7 contarction_not

```
NLP.contarction_not = re.compile(r'n\'t')
```

5.3.3.8 contarction_will

```
NLP.contarction_will = re.compile(r'\'ll')
```

5.3.3.9 d

```
NLP.d = re.compile(r'(?:\b(?:\)|\|\)|\[\)|\?|\|>\|o))')
```

5.3.3.10 database

```
NLP.database = DB()
```

5.3.3.11 e

```
NLP.e = re.compile(r'(?:\b(?:&|€|\[-))')
```

5.3.3.12 f

```
NLP.f = re.compile(r'(?:\b(?:\|=|/=|ph|ƒ|\|#))')
```

5.3.3.13 g

```
NLP.g = re.compile(r'(?:\b(?:&|C-|\(_\+))')
```

5.3.3.14 h

```
NLP.h = re.compile(r'(?:\b(?:#|}{|\|-\||\]-\[\|\[-\]|\(-\)|\)-\(|/-/))')
```

5.3.3.15 hate

```
dictionary NLP.hate = {i[0] for i in database.db_load_lexicon()}
```

5.3.3.16 i

```
NLP.i = re.compile(r'(?:\b(?:!|\||\]|¡))')
```

5.3.3.17 j

```
NLP.j = re.compile(r'(?:\b(?:¿|_\||_/|</|\(/))')
```

5.3.3.18 k

```
NLP.k = re.compile(r'(?:\b(?:\|<|\|\{|\|\()))')
```

5.3.3.19 known_words

```
NLP.known_words = set(words.words())
```

5.3.3.20 letter_l

```
NLP.letter_l = re.compile(r'(?:\b(?:\||£|\|_|¬))')
```

5.3.3.21 m

```
NLP.m = re.compile(r'(?:\b(?:\|v\||\|\\/\||/\\/\\|\(v\)|/\|\\|//\.|\^\^))')
```

5.3.3.22 maxsize

```
NLP.maxsize
```

5.3.3.23 mention_hashtag_regex

```
NLP.mention_hashtag_regex = re.compile(r'([^\w]|^)@[\w\-]+|^[\s]#([^\s])+')
```

5.3.3.24 n

```
NLP.n = re.compile(r'(?:\b(?:\|\\\||/\\/|\[\\\]|<\\>|/V|\^/))')
```

5.3.3.25 o

```
NLP.o = re.compile(r'(?:\b(?:\(\)|\[\]|°))')
```

5.3.3.26 p

```
NLP.p = re.compile(r'(?:\b(?:\|\*|¶|\|o|\|°|\|\"|\|>|\|\|\^\(o\)|\|\|\^\(\)))')
```

5.3.3.27 q

```
NLP.q = re.compile(r'(?:\b(?:\(\)_|\(_,\)|<\|))')
```

5.3.3.28 r

```
NLP.r = re.compile(r'(?:\b(?:\|\^|\|\?|®))')
```

5.3.3.29 reg

```
NLP.reg = re.compile(r'(.)\1{2,}')
```

5.3.3.30 s

```
NLP.s = re.compile(r'(?:\b(?:\$|§))')
```

5.3.3.31 stopwords_set

```
NLP.stopwords_set = stopwords.words("english")
```

5.3.3.32 t

```
NLP.t = re.compile(r'(?:\b(?:\+|-\|-|†|\'\[\]\'))')
```

5.3.3.33 tag

```
NLP.tag = nltk.pos_tag
```

5.3.3.34 tokenize

```
NLP.tokenize = CustomTweetTokenizer().tokenize
```

5.3.3.35 u

```
NLP.u = re.compile(r'(?:\b(?:\|_\||\(_\)))')
```

5.3.3.36 url_remove

```
NLP.url_remove
```

Initial value:

```
1 =  re.compile(
2
       r'(h(\s)*t(\s)*t(\s)*p(\s)*s?://)(\s)*(www\.)?(\s)*((\w|\s)+\.)*([\w\-\s]+/)*([\w\-]+)((\?)?[\w\s]*=\s*['
3     r'\w%&]*)*')
```

5.3.3.37 v

```
NLP.v = re.compile(r'(?:\b(?:\\/|\^))')
```

5.3.3.38 w

```
NLP.w = re.compile(r'(?:\b(?:VV|\\/\\/|\\\\\'|\'//|\\\|/|\\\^/))')
```

5.3.3.39 wnl

```
NLP.wnl = WordNetLemmatizer()
```

5.3.3.40 x

```
NLP.x = re.compile(r'(?:\b(?:><|\)\(|%))')
```

5.3.3.41 y

```
NLP.y = re.compile(r'(?:\b(?:¥|\'/))')
```

5.3.3.42 z

```
NLP.z = re.compile(r'(?:\b(?:～/_|-/_|>_))')
```

5.4 plot_confusion_matrix Namespace Reference

Functions

- def plot ()
- def accaracy ()

5.4.1 Function Documentation

5.4.1.1 accaracy()

```
def plot_confusion_matrix.accaracy ( )
```

```
  to recalculate accuracy
```

5.4.1.2 plot()

```
def plot_confusion_matrix.plot ( )
```

```
  plot a confusion matrix
```

5.5 server Namespace Reference

Functions

- def analyze ()
- def process ()
- def init ()
- def showdata ()
- def showhate ()
- def initmodel ()
- def initmodelsmall ()
- def initmodelmedium ()
- def statusmodel ()

Variables

- app = Flask(__name__)
- database = DB()
- list tweets = [i[0] for i in database.db_load_tweet()]
- list hate = [i[0] for i in database.db_load_hate()]
- methods

5.5.1 Detailed Description

```
@package server
Provides the api routes and calls ai functions. With this we can implement an webinterface.

More details.
```

5.5.2 Function Documentation

5.5.2.1 analyze()

```
def server.analyze ( )

  analyse a tweet using logistic word model
```

5.5.2.2 init()

```
def server.init ( )

  init the server and program

download nesseary files for nlp
```

5.5.2.3 initmodel()

```
def server.initmodel ( )

  init a model

modelname is given by the url parameter "modelname"
```

5.5.2.4 initmodelmedium()

```
def server.initmodelmedium ( )

  init a model but with only a 1000 tweets

modelname is given by the url parameter "modelname"
for debugging
```

5.5.2.5 initmodelsmall()

```
def server.initmodelsmall ( )
```

```
  init a model but with only a 100 tweets

modelname is given by the url parameter "modelname"
for debugging
```

5.5.2.6 process()

```
def server.process ( )
```

```
  Get a preprocesing of a tweet

for debugging
```

5.5.2.7 showdata()

```
def server.showdata ( )
```

```
  give tweets present in the tweets database
```

5.5.2.8 showhate()

```
def server.showhate ( )
```

```
  give tweets present in the tweets database
```

5.5.2.9 statusmodel()

```
def server.statusmodel ( )
```

```
  give back if model is done building.
```

5.5.3 Variable Documentation

5.5.3.1 app

```
server.app = Flask(__name__)
```

5.5.3.2 database

```
server.database = DB()
```

5.5.3.3 hate

```
list server.hate = [i[0] for i in database.db_load_hate()]
```

5.5.3.4 methods

```
server.methods
```

5.5.3.5 tweets

```
list server.tweets = [i[0] for i in database.db_load_tweet()]
```

5.6 wsgi Namespace Reference

5.6.1 Detailed Description

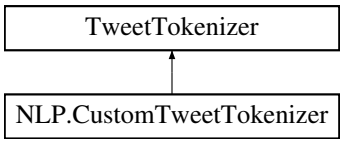
```
@package wsgi
entry for Gunicorn to run the flask server as a service on the server.
```

Chapter 6

Class Documentation

6.1 NLP.CustomTweetTokenizer Class Reference

Inheritance diagram for NLP.CustomTweetTokenizer:



Public Member Functions

- def tokenize (self, text)

6.1.1 Member Function Documentation

6.1.1.1 tokenize()

```
def NLP.CustomTweetTokenizer.tokenize (
              self,
              text )
```

The documentation for this class was generated from the following file:

- NLP.py

6.2 db.DB Class Reference

Public Member Functions

- def __init__ (self)
- def refresh_token (self)
- def expired (self)
- def create_lexicon_db (self)
- def show_lexicon_db (self)
- def create_data_db (self)
- def create_adversarial_db (self)
- def create_extra_db (self)
- def show_data_db (self)
- def db_load_lexicon (self)
- def db_load_tweet (self)
- def db_load_hate (self)
- def db_load_ad_hate (self)
- def db_load_ad_tweet (self)
- def db_load_extra_hate (self)
- def db_load_extra_tweet (self)
- def create_model_db (self)
- def model_in_db (self, name)
- def insert_model_in_db (self, name, model)
- def insert_vect_in_db (self, name, vect)
- def constructing_model_in_db (self, name)
- def get_model_in_db (self, name)

Static Public Member Functions

- def insert_term (i, cursor)
- def insert_ad (i, cursor)
- def insert_extra (i, cursor)
- def insert_data (i, cursor)

Public Attributes

- conn_lexicon
- conn_data
- conn_model
- conn_ad
- conn_extra_data
- token
- expires

6.2.1 Detailed Description

```
collection of all db connections
```

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def db.DB.__init__ (
              self )
```

6.2.3 Member Function Documentation

6.2.3.1 constructing_model_in_db()

```
def db.DB.constructing_model_in_db (
              self,
              name )
```

construct a model, remove existing model in db

to evade predicting when a model is rebuilding

6.2.3.2 create_adversarial_db()

```
def db.DB.create_adversarial_db (
              self )
```

Create the adversarial database

Used to create adversarials totally random

6.2.3.3 create_data_db()

```
def db.DB.create_data_db (
              self )
```

Read the csv and create the data database

6.2.3.4 create_extra_db()

```
def db.DB.create_extra_db (
              self )
```

create a data database using a new diffrent dataset

6.2.3.5 create_lexicon_db()

```
def db.DB.create_lexicon_db (
              self )
```

create and populate the lexicon database

6.2.3.6 create_model_db()

```
def db.DB.create_model_db (
              self )
```

Create a model database

6.2.3.7 db_load_ad_hate()

```
def db.DB.db_load_ad_hate (
              self )
```

give an array of all hate labels in adversarial database

6.2.3.8 db_load_ad_tweet()

```
def db.DB.db_load_ad_tweet (
              self )
```

give an array of all tweets in adversarial database

6.2.3.9 db_load_extra_hate()

```
def db.DB.db_load_extra_hate (
              self )
```

give an array of all hate labels in extra data database

6.2.3.10 db_load_extra_tweet()

```
def db.DB.db_load_extra_tweet (
              self )
```

give an array of all tweets in extra data database

6.2.3.11 db_load_hate()

```
def db.DB.db_load_hate (
              self )
```

give an array of all hate labels in data

6.2.3.12 db_load_lexicon()

```
def db.DB.db_load_lexicon (
              self )
```

give an array of all term in lexicon

6.2.3.13 db_load_tweet()

```
def db.DB.db_load_tweet (
              self )
```

give an array of all tweets in data

6.2.3.14 expired()

```
def db.DB.expired (
              self )
```

check if hatebase token is expired

6.2.3.15 get_model_in_db()

```
def db.DB.get_model_in_db (
              self,
              name )
```

get a model from the model database

6.2.3.16 insert_ad()

```
def db.DB.insert_ad (
              i,
              cursor )  [static]
```

insert an adversarial into the adversarial database

6.2.3.17 insert_data()

```
def db.DB.insert_data (
              i,
              cursor )  [static]
```

insert an term into the data database

6.2.3.18 insert_extra()

```
def db.DB.insert_extra (
              i,
              cursor )  [static]
```

insert a tweet into the extra data database

6.2.3.19 insert_model_in_db()

```
def db.DB.insert_model_in_db (
              self,
              name,
              model )
```

insert a model in the model database

6.2.3.20 insert_term()

```
def db.DB.insert_term (
              i,
              cursor )  [static]
```

insert a term in the lexicon database

6.2.3.21 insert_vect_in_db()

```
def db.DB.insert_vect_in_db (
              self,
              name,
              vect )
```

insert a vectorizer in the model database

6.2.3.22 model_in_db()

```
def db.DB.model_in_db (
              self,
              name )
```

check if a model is in the model database

6.2.3.23 refresh_token()

```
def db.DB.refresh_token (
              self )
```

refresh hatebase token

6.2.3.24 show_data_db()

```
def db.DB.show_data_db (
              self )
```

print data database

6.2.3.25 show_lexicon_db()

```
def db.DB.show_lexicon_db (
              self )
```

print the lexicon database

6.2.4 Member Data Documentation

6.2.4.1 conn_ad

```
db.DB.conn_ad
```

6.2.4.2 conn_data

```
db.DB.conn_data
```

6.2.4.3 conn_extra_data

```
db.DB.conn_extra_data
```

6.2.4.4 conn_lexicon

db.DB.conn_lexicon

6.2.4.5 conn_model

db.DB.conn_model

6.2.4.6 expires

db.DB.expires

6.2.4.7 token

db.DB.token

The documentation for this class was generated from the following file:

- db.py

Chapter 7

File Documentation

7.1 ai.py File Reference

Namespaces

- ai

Functions

- def ai.analyse_text (text, modelname="logistic_regression")
- def ai.analyse_ad_lstm (modelname)
- def ai.analyse_ad ()
- def ai.process_text (text)
- def ai.return_token (text)
- def ai.construct_model (data, hate, modelname="logistic_regression")
- def ai.logistic (vectorizer, data, hate, modelname)
- def ai.parallel_construct (data, func)
- def ai.naive_bayes (vectorizer, data, hate, modelname)
- def ai.construct_lstm (data, hate, tokenizer, modelname, maxlen=500)
- def ai.construct_lstm_tibo (data, hate, modelname)
- def ai.construct_lstm_les (data, hate, tokenizer, modelname, maxlen=500)
- def ai.make_lstm_model (x)
- def ai.make_lstm_les_model (x)

7.2 db.py File Reference

Classes

- class db.DB

Namespaces

- db

7.3 NLP.py File Reference

Classes

- class NLP.CustomTweetTokenizer

Namespaces

- NLP

Functions

- def NLP.lemmatize (token, pos)
- def NLP.wordsegment (token)
- def NLP.spell_checker (token)
- def NLP.text_precessing (text)
- def NLP.basic_precessing (text)
- def NLP.basic_precessing_char (text)
- def NLP.text_precessing_char (text)
- def NLP.remove_repeats (word)
- def NLP.get_wordnet_pos (treebank_tag)
- def NLP.char_boundary (tokens)
- def NLP.has_word (word, voca)
- def NLP.lemmanize_text (tokens)

Variables

- NLP.reg = re.compile(r'(.)\1{2,}')
- NLP.mention_hashtag_regex = re.compile(r'([^\w]|^)@[\w\-]+|^[\s]#([^\s])+')
- NLP.url_remove
- NLP.contarction_not = re.compile(r'n\'t')
- NLP.contarction_am = re.compile(r'\'m')
- NLP.contarction_have = re.compile(r'\'ve')
- NLP.contarction_will = re.compile(r'\'ll')
- NLP.a = re.compile(r'(?:\b(?:@|/-\\|\^|/\\))')
- NLP.b = re.compile(r'(?:\b(?:\|:|P>|ß))')
- NLP.c = re.compile(r'(?:\b[©¢<\[({])')
- NLP.d = re.compile(r'(?:\b(?:\)|\|\)|\[\)|\?|\|>\|o))')
- NLP.e = re.compile(r'(?:\b(?:&|€|\[-))')
- NLP.f = re.compile(r'(?:\b(?:\|=|/=|ph|ƒ|\|#))')
- NLP.g = re.compile(r'(?:\b(?:&|C-|\(_\+))')
- NLP.h = re.compile(r'(?:\b(?:#|}{|\|-\||\]-\[\|\[-\]|\(-\)|\)-\(|/-/))')
- NLP.i = re.compile(r'(?:\b(?:!|\||\]|¡))')
- NLP.j = re.compile(r'(?:\b(?:¿|_\||_/|</|\(/))')
- NLP.k = re.compile(r'(?:\b(?:\|<|\|\{|\|\()')
- NLP.letter_l = re.compile(r'(?:\b(?:\||£|\|_|¬))')
- NLP.m = re.compile(r'(?:\b(?:\|v\||\|\/\\||/\\/\\|\(v\)|/\|\\|/\.|\^\^))')
- NLP.n = re.compile(r'(?:\b(?:\|\\\||/\\/|\[\\\]|<\\>|/V|\^/))')
- NLP.o = re.compile(r'(?:\b(?:\(\)|\[\]|°))')
- NLP.p = re.compile(r'(?:\b(?:\|*|¶|\|o|\|°|\|\"|\|>|\|\^\(o\)|\|\^\(\)))')
- NLP.q = re.compile(r'(?:\b(?:\(\)_|\(_,\)|<\|))')

- NLP.r = re.compile(r'(?:\b(?:\|\^|\|\?|®))')
- NLP.s = re.compile(r'(?:\b(?:\$|§))')
- NLP.t = re.compile(r'(?:\b(?:\+|-\|-|†|\'\[\]\')))')
- NLP.u = re.compile(r'(?:\b(?:\|_\||\(_\)))')
- NLP.v = re.compile(r'(?:\b(?:\V|\^))')
- NLP.w = re.compile(r'(?:\b(?:VV|\\/\\/|\\\\\'|\'//|\\\|/|\\\^/))')
- NLP.x = re.compile(r'(?:\b(?:><|\)\(|%))')
- NLP.y = re.compile(r'(?:\b(?:¥|\'/))')
- NLP.z = re.compile(r'(?:\b(?:~/_|-/_|>_))')
- NLP.checker = SpellChecker()
- NLP.wnl = WordNetLemmatizer()
- NLP.tag = nltk.pos_tag
- NLP.stopwords_set = stopwords.words("english")
- NLP.known_words = set(words.words())
- NLP.database = DB()
- dictionary NLP.hate = {i[0] for i in database.db_load_lexicon()}
- NLP.maxsize
- NLP.tokenize = CustomTweetTokenizer().tokenize

7.4 plot_confusion_matrix.py File Reference

Namespaces

- plot_confusion_matrix

Functions

- def plot_confusion_matrix.plot ()
- def plot_confusion_matrix.accaracy ()

7.5 requirements.txt File Reference

7.6 server.py File Reference

Namespaces

- server

Functions

- def server.analyze ()
- def server.process ()
- def server.init ()
- def server.showdata ()
- def server.showhate ()
- def server.initmodel ()
- def server.initmodelsmall ()
- def server.initmodelmedium ()
- def server.statusmodel ()

Variables

- server.app = Flask(__name__)
- server.database = DB()
- list server.tweets = [i[0] for i in database.db_load_tweet()]
- list server.hate = [i[0] for i in database.db_load_hate()]
- server.methods

7.7 wsgi.py File Reference

Namespaces

- wsgi

Index