

MakeUHappy Chat-bot

Создано системой Doxygen 1.8.20

1 Алфавитный указатель пространств имен	1
1.1 Пространства имен	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Пространства имен	7
4.1 Пространство имен <code>bad_language</code>	7
4.2 Пространство имен <code>dialogpt</code>	7
4.2.1 Подробное описание	7
4.3 Пространство имен <code>embeddings</code>	7
4.4 Пространство имен <code>JokeClassifier</code>	8
4.5 Пространство имен <code>prod</code>	8
4.5.1 Подробное описание	8
4.5.2 Переменные	8
4.5.2.1 <code>args</code>	8
4.5.2.2 <code>help</code>	8
4.5.2.3 <code>parser</code>	9
4.5.2.4 <code>required</code>	9
4.5.2.5 <code>True</code>	9
4.6 Пространство имен <code>prod.worker</code>	9
4.6.1 Подробное описание	9
4.7 Пространство имен <code>text</code>	9
4.7.1 Подробное описание	10
4.7.2 Функции	10
4.7.2.1 <code>extract_text()</code>	10
4.7.2.2 <code>get_joke()</code>	10
4.7.2.3 <code>get_text()</code>	10
4.8 Пространство имен <code>translate_emoji</code>	11
4.8.1 Подробное описание	11
4.8.2 Функции	11
4.8.2.1 <code>de_emojify()</code>	11
4.8.2.2 <code>emoji_to_text()</code>	12
4.8.2.3 <code>eng_to_rus()</code>	12
4.8.3 Переменные	13
4.8.3.1 <code>translator</code>	13
4.9 Пространство имен <code>web</code>	13
4.9.1 Подробное описание	13
4.9.2 Функции	13
4.9.2.1 <code>get_advice()</code>	14
4.9.2.2 <code>get_updates()</code>	14

4.9.2.3 <code>send_message()</code>	14
4.9.3 Переменные	15
4.9.3.1 <code>api_url</code>	15
5 Классы	17
5.1 Класс <code>dialogpt.DialoGPT</code>	17
5.1.1 Подробное описание	18
5.1.2 Конструктор(ы)	18
5.1.2.1 <code>__init__()</code>	18
5.1.3 Методы	18
5.1.3.1 <code>generate()</code>	18
5.1.3.2 <code>get_length_param()</code>	19
5.1.3.3 <code>get_response()</code>	19
5.1.3.4 <code>process_text()</code>	20
5.1.3.5 <code>restart()</code>	20
5.1.4 Данные класса	20
5.1.4.1 <code>chat_history</code>	20
5.1.4.2 <code>model</code>	20
5.1.4.3 <code>tokenizer</code>	21
5.1.4.4 <code>user_input_size</code>	21
5.1.4.5 <code>window_size</code>	21
5.2 Класс <code>embeddings.EmbeddingCreator</code>	21
5.2.1 Подробное описание	22
5.2.2 Конструктор(ы)	22
5.2.2.1 <code>__init__()</code>	22
5.2.3 Методы	22
5.2.3.1 <code>get_by_index()</code>	22
5.2.3.2 <code>get_phrase_embedding()</code>	22
5.2.3.3 <code>make_embeddings()</code>	23
5.2.3.4 <code>run()</code>	23
5.2.3.5 <code>tokenize()</code>	23
5.2.3.6 <code>write_matrix()</code>	23
5.2.4 Данные класса	23
5.2.4.1 <code>device</code>	23
5.2.4.2 <code>embeddings</code>	24
5.2.4.3 <code>model</code>	24
5.2.4.4 <code>tokenized</code>	24
5.2.4.5 <code>tokenizer</code>	24
5.2.4.6 <code>words</code>	24
5.3 Класс <code>JokeClassifier</code>	24
5.3.1 Подробное описание	25
5.4 Класс <code>JokeClassifier.JokeClassifier</code>	25
5.4.1 Подробное описание	25

5.4.2	Конструктор(ы)	25
5.4.2.1	<code>__init__()</code>	25
5.4.3	Методы	26
5.4.3.1	<code>__call__()</code>	26
5.4.4	Данные класса	26
5.4.4.1	<code>model</code>	26
5.4.4.2	<code>tokenizer</code>	26
5.5	Класс <code>bad_language.Merger</code>	27
5.5.1	Подробное описание	27
5.5.2	Конструктор(ы)	27
5.5.2.1	<code>__init__()</code>	27
5.5.3	Методы	27
5.5.3.1	<code>__call__()</code>	27
5.5.4	Данные класса	27
5.5.4.1	<code>classifiers</code>	28
5.5.4.2	<code>is_soft</code>	28
5.6	Класс <code>embeddings.QuestionFinder</code>	28
5.6.1	Подробное описание	28
5.6.2	Конструктор(ы)	28
5.6.2.1	<code>__init__()</code>	28
5.6.3	Методы	29
5.6.3.1	<code>find_closest()</code>	29
5.6.4	Данные класса	29
5.6.4.1	<code>dim</code>	29
5.6.4.2	<code>embeddings</code>	29
5.6.4.3	<code>labels</code>	29
5.6.4.4	<code>num</code>	29
5.6.4.5	<code>p</code>	30
5.7	Класс <code>bad_language.SwearDetector</code>	30
5.7.1	Подробное описание	30
5.7.2	Конструктор(ы)	30
5.7.2.1	<code>__init__()</code>	30
5.7.3	Методы	30
5.7.3.1	<code>__call__()</code>	31
5.7.3.2	<code>clear()</code>	31
5.7.3.3	<code>find_swear()</code>	31
5.7.3.4	<code>has_swear()</code>	31
5.7.4	Данные класса	31
5.7.4.1	<code>blocklist</code>	31
5.7.4.2	<code>lemmatizer</code>	32
5.8	Класс <code>bad_language.ToxicClassifier</code>	32
5.8.1	Подробное описание	32
5.8.2	Конструктор(ы)	32

5.8.2.1 <code>__init__()</code>	32
5.8.3 Методы	32
5.8.3.1 <code>__call__()</code>	33
5.8.3.2 <code>how_toxic()</code>	33
5.8.3.3 <code>is_toxic()</code>	33
5.8.4 Данные класса	33
5.8.4.1 <code>toxic_bert</code>	33
5.8.4.2 <code>toxic_tokenizer</code>	33
5.9 Класс <code>prod.worker.Worker</code>	34
5.9.1 Подробное описание	34
5.9.2 Конструктор(ы)	34
5.9.2.1 <code>__init__()</code>	34
5.9.3 Методы	35
5.9.3.1 <code>work()</code>	35
5.9.3.2 <code>work_once()</code>	35
5.9.4 Данные класса	35
5.9.4.1 <code>chat_bots</code>	35
5.9.4.2 <code>offset</code>	35
5.9.4.3 <code>token</code>	35
6 Файлы	37
6.1 Файл <code>__init__.py</code>	37
6.2 Файл <code>utils/bad_language.py</code>	37
6.3 Файл <code>utils/embeddings.py</code>	37
6.4 Файл <code>utils/JokeClassifier.py</code>	38
6.5 Файл <code>utils/telegram/dialogpt.py</code>	38
6.6 Файл <code>utils/telegram/text.py</code>	38
6.7 Файл <code>utils/telegram/translate_emoji.py</code>	39
6.8 Файл <code>utils/telegram/web.py</code>	39
6.9 Файл <code>worker.py</code>	39
Предметный указатель	41

Глава 1

Алфавитный указатель пространств имен

1.1 Пространства имен

Полный список пространств имен.

bad_language	7
dialogpt	
Содержит класс чат-бота DialoGPT	7
embeddings	7
JokeClassifier	8
prod	
Содержит всё для работы Telegram бота	8
prod.worker	
Содержит класс Worker	9
text	
Содержит функции для работы с текстами и с Telegram	9
translate_emoji	
Содержит материалы для обработки эмоджи	11
web	
Содержит функции для взаимодействия с веб-сервисами	13

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

dialogpt.DialoGPT	
Класс, содержащий в себе интерфейс работы с DialoGPT	17
embeddings.EmbeddingCreator	21
JokeClassifier	
Класс для определения является ли сообщение шуткой	24
JokeClassifier.JokeClassifier	25
bad_language.Merger	27
embeddings.QuestionFinder	28
bad_language.SwearDetector	30
bad_language.ToxicClassifier	32
prod.worker.Worker	
Класс для работы с Telegram API	34

Глава 3

Список файлов

3.1 Файлы

Полный список файлов.

__init__.py	37
worker.py	39
utils/bad_language.py	37
utils/embeddings.py	37
utils/JokeClassifier.py	38
utils/telegram/dialogpt.py	38
utils/telegram/text.py	38
utils/telegram/translate_emoji.py	39
utils/telegram/web.py	39

Глава 4

Пространства имен

4.1 Пространство имен `bad_language`

Классы

- class [Merger](#)
- class [SwearDetector](#)
- class [ToxicClassifier](#)

4.2 Пространство имен `dialogpt`

Содержит класс чат-бота [DialoGPT](#).

Классы

- class [DialoGPT](#)
Класс, содержащий в себе интерфейс работы с [DialoGPT](#).

4.2.1 Подробное описание

Содержит класс чат-бота [DialoGPT](#).

4.3 Пространство имен `embeddings`

Классы

- class [EmbeddingCreator](#)
- class [QuestionFinder](#)

4.4 Пространство имен JokeClassifier

Классы

- class [JokeClassifier](#)

4.5 Пространство имен prod

Содержит всё для работы Telegram бота

Пространства имен

- [worker](#)
Содержит класс [Worker](#).

Переменные

- [parser](#) = argparse.ArgumentParser()
- [required](#)
- [True](#)
- [help](#)
- [args](#) = parser.parse_args()

4.5.1 Подробное описание

Содержит всё для работы Telegram бота

4.5.2 Переменные

4.5.2.1 args

```
prod.args = parser.parse_args()
```

См. определение в файле `__init__.py` строка 10

4.5.2.2 help

```
prod.help
```

См. определение в файле `__init__.py` строка 9

4.5.2.3 parser

```
prod.parser = argparse.ArgumentParser()
```

См. определение в файле `__init__.py` строка 8

4.5.2.4 required

```
prod.required
```

См. определение в файле `__init__.py` строка 9

4.5.2.5 True

```
prod.True
```

См. определение в файле `__init__.py` строка 9

4.6 Пространство имен prod.worker

Содержит класс [Worker](#).

Классы

- class [Worker](#)
Класс для работы с Telegram API.

4.6.1 Подробное описание

Содержит класс [Worker](#).

4.7 Пространство имен text

Содержит функции для работы с текстами и с Telegram.

Функции

- def [extract_text](#) (message)
Извлекает текст из сообщения
- def [get_text](#) (message)
Получает и обрабатывает текст из сообщения
- def [get_joke](#) ()
Выводит случайный анекдот из набора случайных анекдотов

4.7.1 Подробное описание

Содержит функции для работы с текстами и с Telegram.

4.7.2 Функции

4.7.2.1 `extract_text()`

```
def text.extract_text (
    message )
```

Извлекает текст из сообщения

Аргументы

<code>message</code>	Получаемое сообщение из Telegram API
----------------------	--------------------------------------

Возвращает

Текст из сообщения

См. определение в файле `text.py` строка 10

4.7.2.2 `get_joke()`

```
def text.get_joke ( )
```

Выводит случайный анекдот из набора случайных анекдотов

Возвращает

Случайный анекдот

См. определение в файле `text.py` строка 30

4.7.2.3 `get_text()`

```
def text.get_text (
    message )
```

Получает и обрабатывает текст из сообщения

Аргументы

<code>message</code>	Получаемое сообщение из Telegram API
----------------------	--------------------------------------

Возвращает

Обработанный текст

См. определение в файле `text.py` строка 24

4.8 Пространство имен `translate_emoji`

Содержит материалы для обработки эмоджи

Функции

- `def eng_to_rus (text)`
Переводит текст с английского на русский
- `def emoji_to_text (emoji)`
Переводит эмоджи в текст на русском
- `def de_emojify (text)`
Переводит эмоджи в тексте на русский язык

Переменные

- `translator = google_translator()`
Переводчик

4.8.1 Подробное описание

Содержит материалы для обработки эмоджи

4.8.2 Функции

4.8.2.1 `de_emojify()`

```
def translate_emoji.de_emojify (  
    text )
```

Переводит эмоджи в тексте на русский язык

См. также

<https://unicode.org/emoji/charts/full-emoji-list.html>

Аргументы

text	Текст на русском
------	------------------

Возвращает

Текст на русском без эмоджи

См. определение в файле `translate_emoji.py` строка 33

4.8.2.2 emoji_to_text()

```
def translate_emoji.emoji_to_text (
    emoji )
```

Переводит эмоджи в текст на русском

См. также

<https://unicode.org/emoji/charts/full-emoji-list.html>

Аргументы

emoji	Эмоджи
-------	--------

Возвращает

Значение эмоджи на русском

См. определение в файле `translate_emoji.py` строка 22

4.8.2.3 eng_to_rus()

```
def translate_emoji.eng_to_rus (
    text )
```

Переводит текст с английского на русский

Аргументы

text	Текст на английском
------	---------------------

Возвращает

Текст на русском

См. определение в файле `translate_emoji.py` строка 15

4.8.3 Переменные

4.8.3.1 translator

```
translate_emoji.translator = google_translator()
```

Переводчик

См. определение в файле `translate_emoji.py` строка 10

4.9 Пространство имен web

Содержит функции для взаимодействия с веб-сервисами

Функции

- `def get_updates (token, offset)`
Получает новые сообщения с сервера.
- `def send_message (token, chat_id, text)`
Отправляет сообщение в чат.
- `def get_advice ()`
Достаёт случайный совет

Переменные

- `string api_url = 'https://api.telegram.org/'`
Ссылка на Telegram API.

4.9.1 Подробное описание

Содержит функции для взаимодействия с веб-сервисами

4.9.2 Функции

4.9.2.1 get_advice()

```
def web.get_advice ( )
```

Достаёт случайный совет

См. также

<https://fucking-great-advice.ru/api>

Возвращает

Случайный совет в виде строки

См. определение в файле web.py строка 35

4.9.2.2 get_updates()

```
def web.get_updates (
    token,
    offset )
```

Получает новые сообщения с сервера.

Если сообщений нет, то возвращает пустой массив.

См. также

<https://core.telegram.org/bots/api#getupdates>

Аргументы

token	Токен для чат-бота
offset	Параметр offset в Telegram API

Возвращает

Массив с последними сообщениями

См. определение в файле web.py строка 17

4.9.2.3 send_message()

```
def web.send_message (
    token,
```

```
chat_id,  
text )
```

Отправляет сообщение в чат.

См. также

<https://core.telegram.org/bots/api#sendmessage>

Аргументы

token	Токен для чат-бота
chat↔ _id	id чата
text	Отправляемое сообщение

См. определение в файле web.py строка 27

4.9.3 Переменные

4.9.3.1 api_url

```
web.api_url = 'https://api.telegram.org/'
```

Ссылка на Telegram API.

См. определение в файле web.py строка 8

Глава 5

Классы

5.1 Класс `dialogpt.DialogPT`

Класс, содержащий в себе интерфейс работы с [DialogPT](#).

Открытые члены

- `def __init__(self, window_size=10)`
Создаёт объект класса [DialogPT](#) для отдельного чата
- `str get_length_param(self, str text)`
- `def generate(self, bot_input_ids, num_return_sequences=1, max_length=512, no_repeat_ngram_size=3, do_sample=True, top_k=50, top_p=0.9, temperature=0.6, mask_token_id=tokenizer.mask_token_id, eos_token_id=tokenizer.eos_token_id, unk_token_id=tokenizer.unk_token_id, pad_token_id=tokenizer.pad_token_id, device='cuda')`
Передаёт аргументы в функцию `self.model.generate` и вызывает её
- `def process_text(self, input_user)`
Обрабатывает текст без команды
- `def get_response(self, text)`
Обрабатывает текст с возможной командой
- `def restart(self)`
Забывает все предыдущие фразы в диалоге

Открытые атрибуты

- `chat_history`
Содержимое текущего диалога
- `user_input_size`
Размеры каждой фразы диалога
- `window_size`
Число запоминаемых фраз в диалоге

Статические открытые данные

- `tokenizer = AutoTokenizer.from_pretrained("Grossmend/rudialogpt3_medium_based_on_gpt2")`
Токенайзер модели [DialoGPT](#).
- `model = AutoModelForCausalLM.from_pretrained("Grossmend/rudialogpt3_medium_based_on_gpt2")`
Сама модель [DialoGPT](#).

5.1.1 Подробное описание

Класс, содержащий в себе интерфейс работы с [DialoGPT](#).

См. определение в файле `dialogpt.py` строка 11

5.1.2 Конструктор(ы)

5.1.2.1 `__init__()`

```
def dialogpt.DialoGPT.__init__(
    self,
    window_size = 10 )
```

Создаёт объект класса [DialoGPT](#) для отдельного чата

Аргументы

<code>window_size</code>	Число диалогов, которых запоминает бот
--------------------------	--

См. определение в файле `dialogpt.py` строка 22

5.1.3 Методы

5.1.3.1 `generate()`

```
def dialogpt.DialoGPT.generate (
    self,
    bot_input_ids,
    num_return_sequences = 1,
    max_length = 512,
    no_repeat_ngram_size = 3,
    do_sample = True,
    top_k = 50,
```



```

top_p = 0.9,
temperature = 0.6,
mask_token_id = tokenizer.mask_token_id,
eos_token_id = tokenizer.eos_token_id,
unk_token_id = tokenizer.unk_token_id,
pad_token_id = tokenizer.pad_token_id,
device = 'cuda ' )

```

Передаёт аргументы в функцию `self.model.generate` и вызывает её

Аргументы

<code>bot_input_ids</code>	Токены, подаваемые <code>self.model.generate</code> на вход
----------------------------	---

Возвращает

Токены, сгенерированные моделью

См. определение в файле `dialogpt.py` строка 50

5.1.3.2 `get_length_param()`

```

str dialogpt.DialoGPT.get_length_param (
    self,
    str text )

```

См. определение в файле `dialogpt.py` строка 35

5.1.3.3 `get_response()`

```

def dialogpt.DialoGPT.get_response (
    self,
    text )

```

Обрабатывает текст с возможной командой

Аргументы

<code>text</code>	Текст, в котором может содержаться команда
-------------------	--

Возвращает

Итоговый текст

См. определение в файле `dialogpt.py` строка 109

5.1.3.4 process_text()

```
def dialogpt.DialoGPT.process_text (
    self,
    input_user )
```

Обрабатывает текст без команды

Аргументы

input_user	Текст без команды
------------	-------------------

Возвращает

Ответ на текст

См. определение в файле dialogpt.py строка 82

5.1.3.5 restart()

```
def dialogpt.DialoGPT.restart (
    self )
```

Забывает все предыдущие фразы в диалоге

См. определение в файле dialogpt.py строка 144

5.1.4 Данные класса

5.1.4.1 chat_history

```
dialogpt.DialoGPT.chat_history
```

Содержимое текущего диалога

См. определение в файле dialogpt.py строка 25

5.1.4.2 model

```
dialogpt.DialoGPT.model = AutoModelForCausalLM.from_pretrained("Grossmend/rudialogpt3_medium_based_on_gpt2") [static]
```

Сама модель [DialoGPT](#).

См. определение в файле dialogpt.py строка 17

5.1.4.3 tokenizer

```
dialogpt.DialoGPT.tokenizer = AutoTokenizer.from_pretrained("Grossmend/rudialogpt3_medium_based_on_gpt2")  
[static]
```

Токенайзер модели [DialoGPT](#).

См. определение в файле dialogpt.py строка 14

5.1.4.4 user_input_size

```
dialogpt.DialoGPT.user_input_size
```

Размеры каждой фразы диалога

См. определение в файле dialogpt.py строка 29

5.1.4.5 window_size

```
dialogpt.DialoGPT.window_size
```

Число запоминаемых фраз в диалоге

См. определение в файле dialogpt.py строка 33

Объявления и описания членов класса находятся в файле:

- [utils/telegram/dialogpt.py](#)

5.2 Класс embeddings.EmbeddingCreator

Открытые члены

- `def __init__ (self, model_name_or_path="sberbank-ai/rugpt3small_based_on_gpt2")`
- `def tokenize (self, data)`
- `def make_embeddings (self, tokenized_data)`
- `def write_matrix (self, data, filename='embeddings')`
- `def run (self, path_or_word_list, output_filename='embeddings')`
- `def get_phrase_embedding (self, phrase)`
- `def get_by_index (self, index)`

Открытые атрибуты

- [words](#)
- [device](#)
- [tokenizer](#)
- [model](#)
- [tokenized](#)
- [embeddings](#)

5.2.1 Подробное описание

См. определение в файле `embeddings.py` строка 7

5.2.2 Конструктор(ы)

5.2.2.1 `__init__()`

```
def embeddings.EmbeddingCreator.__init__(
    self,
    model_name_or_path = "sberbank-ai/rugpt3small_based_on_gpt2" )
```

For usage examples go to `~/junk/run_this_all.ipynb`

См. определение в файле `embeddings.py` строка 8

5.2.3 Методы

5.2.3.1 `get_by_index()`

```
def embeddings.EmbeddingCreator.get_by_index (
    self,
    index )
```

См. определение в файле `embeddings.py` строка 63

5.2.3.2 `get_phrase_embedding()`

```
def embeddings.EmbeddingCreator.get_phrase_embedding (
    self,
    phrase )
```

См. определение в файле `embeddings.py` строка 57

5.2.3.3 make_embeddings()

```
def embeddings.EmbeddingCreator.make_embeddings (
    self,
    tokenized_data )
```

См. определение в файле embeddings.py строка 21

5.2.3.4 run()

```
def embeddings.EmbeddingCreator.run (
    self,
    path_or_word_list,
    output_filename = 'embeddings' )
```

См. определение в файле embeddings.py строка 33

5.2.3.5 tokenize()

```
def embeddings.EmbeddingCreator.tokenize (
    self,
    data )
```

См. определение в файле embeddings.py строка 17

5.2.3.6 write_matrix()

```
def embeddings.EmbeddingCreator.write_matrix (
    self,
    data,
    filename = 'embeddings' )
```

См. определение в файле embeddings.py строка 29

5.2.4 Данные класса

5.2.4.1 device

```
embeddings.EmbeddingCreator.device
```

См. определение в файле embeddings.py строка 13

5.2.4.2 embeddings

`embeddings.EmbeddingCreator.embeddings`

См. определение в файле `embeddings.py` строка 49

5.2.4.3 model

`embeddings.EmbeddingCreator.model`

См. определение в файле `embeddings.py` строка 15

5.2.4.4 tokenized

`embeddings.EmbeddingCreator.tokenized`

См. определение в файле `embeddings.py` строка 45

5.2.4.5 tokenizer

`embeddings.EmbeddingCreator.tokenizer`

См. определение в файле `embeddings.py` строка 14

5.2.4.6 words

`embeddings.EmbeddingCreator.words`

См. определение в файле `embeddings.py` строка 11

Объявления и описания членов класса находятся в файле:

- [utils/embeddings.py](#)

5.3 Класс JokeClassifier

Класс для определения является ли сообщение шуткой

5.3.1 Подробное описание

Класс для определения является ли сообщение шуткой

Объявления и описания членов класса находятся в файле:

- `utils/JokeClassifier.py`

5.4 Класс JokeClassifier.JokeClassifier

Открытые члены

- `def __init__(self, bert_path, tokenizer_path='DeepPavlov/rubert-base-cased')`
Создаёт объект класса `JokeClassifier`.
- `def __call__(self, sentence)`
Выдаёт вероятность наличия шутки во входном сообщении

Открытые атрибуты

- `tokenizer`
Токенайзер
- `model`
Модель классификатора

5.4.1 Подробное описание

См. определение в файле `JokeClassifier.py` строка 6

5.4.2 Конструктор(ы)

5.4.2.1 `__init__()`

```
def JokeClassifier.JokeClassifier.__init__(
    self,
    bert_path,
    tokenizer_path = 'DeepPavlov/rubert-base-cased ' )
```

Создаёт объект класса `JokeClassifier`.

Аргументы

<code>tokenizer_path</code>	Путь к токенайзеру
<code>bert_path</code>	Путь к модели BERT

См. определение в файле JokeClassifier.py строка 10

5.4.3 Методы

5.4.3.1 `__call__()`

```
def JokeClassifier.JokeClassifier.__call__(  
    self,  
    sentence )
```

Выдаёт вероятность наличия шутки во входном сообщении

Аргументы

sentence	Входное предложение
----------	---------------------

Возвращает

Вероятность наличия шутки

См. определение в файле JokeClassifier.py строка 24

5.4.4 Данные класса

5.4.4.1 `model`

JokeClassifier.JokeClassifier.model

Модель классификатора

См. определение в файле JokeClassifier.py строка 16

5.4.4.2 `tokenizer`

JokeClassifier.JokeClassifier.tokenizer

Токенайзер

См. определение в файле JokeClassifier.py строка 13

Объявления и описания членов класса находятся в файле:

- [utils/JokeClassifier.py](#)

5.5 Класс `bad_language.Merger`

Открытые члены

- `def __init__(self, classifiers=[ToxicClassifier\(\), SwearDetector\(\)], is_soft=False)`
- `def __call__(self, sentence)`

Открытые атрибуты

- `classifiers`
- `is_soft`

5.5.1 Подробное описание

См. определение в файле `bad_language.py` строка 63

5.5.2 Конструктор(ы)

5.5.2.1 `__init__()`

```
def bad_language.Merger.__init__(
    self,
    classifiers = [ToxicClassifier\(\), SwearDetector\(\)],
    is_soft = False )
```

См. определение в файле `bad_language.py` строка 64

5.5.3 Методы

5.5.3.1 `__call__()`

```
def bad_language.Merger.__call__(
    self,
    sentence )
```

См. определение в файле `bad_language.py` строка 68

5.5.4 Данные класса

5.5.4.1 classifiers

`bad_language.Merger.classifiers`

См. определение в файле `bad_language.py` строка 65

5.5.4.2 is_soft

`bad_language.Merger.is_soft`

См. определение в файле `bad_language.py` строка 66

Объявления и описания членов класса находятся в файле:

- [utils/bad_language.py](#)

5.6 Класс `embeddings.QuestionFinder`

Открытые члены

- `def __init__(self, embeddings_path_or_matrix)`
- `def find_closest(self, embedding, num_neighbours=1, remoteness=0)`

Открытые атрибуты

- [embeddings](#)
- [num](#)
- [dim](#)
- [labels](#)
- [p](#)

5.6.1 Подробное описание

См. определение в файле `embeddings.py` строка 74

5.6.2 Конструктор(ы)

5.6.2.1 `__init__()`

```
def embeddings.QuestionFinder.__init__(
    self,
    embeddings_path_or_matrix )
```

For usage examples go to `~/junk/run_this_all.ipynb`

См. определение в файле `embeddings.py` строка 75

5.6.3 Методы

5.6.3.1 `find_closest()`

```
def embeddings.QuestionFinder.find_closest (
    self,
    embedding,
    num_neighbours = 1,
    remoteness = 0 )
```

См. определение в файле `embeddings.py` строка 104

5.6.4 Данные класса

5.6.4.1 `dim`

`embeddings.QuestionFinder.dim`

См. определение в файле `embeddings.py` строка 87

5.6.4.2 `embeddings`

`embeddings.QuestionFinder.embeddings`

См. определение в файле `embeddings.py` строка 77

5.6.4.3 `labels`

`embeddings.QuestionFinder.labels`

См. определение в файле `embeddings.py` строка 89

5.6.4.4 `num`

`embeddings.QuestionFinder.num`

См. определение в файле `embeddings.py` строка 86

5.6.4.5 p

`embeddings.QuestionFinder.p`

См. определение в файле `embeddings.py` строка 91

Объявления и описания членов класса находятся в файле:

- `utils/embeddings.py`

5.7 Класс `bad_language.SwearDetector`

Открытые члены

- `def __init__ (self, path_or_list='../data/swear.txt', use_stemming=True)`
- `def __call__ (self, sentence)`
- `def clear (self, sentence)`
- `def has_swear (self, sentence)`
- `def find_swear (self, sentence)`

Открытые атрибуты

- `lemmatizer`
- `blocklist`

5.7.1 Подробное описание

См. определение в файле `bad_language.py` строка 23

5.7.2 Конструктор(ы)

5.7.2.1 `__init__()`

```
def bad_language.SwearDetector.__init__ (
    self,
    path_or_list = '../data/swear.txt ',
    use_stemming = True )
```

См. определение в файле `bad_language.py` строка 24

5.7.3 Методы

5.7.3.1 __call__()

```
def bad_language.SwearDetector.__call__(  
    self,  
    sentence )
```

См. определение в файле bad_language.py строка 37

5.7.3.2 clear()

```
def bad_language.SwearDetector.clear(  
    self,  
    sentence )
```

См. определение в файле bad_language.py строка 40

5.7.3.3 find_swear()

```
def bad_language.SwearDetector.find_swear(  
    self,  
    sentence )
```

См. определение в файле bad_language.py строка 52

5.7.3.4 has_swear()

```
def bad_language.SwearDetector.has_swear(  
    self,  
    sentence )
```

См. определение в файле bad_language.py строка 43

5.7.4 Данные класса

5.7.4.1 blocklist

```
bad_language.SwearDetector.blocklist
```

См. определение в файле bad_language.py строка 29

5.7.4.2 lemmatizer

`bad_language.SwearDetector.lemmatizer`

См. определение в файле `bad_language.py` строка 25

Объявления и описания членов класса находятся в файле:

- [utils/bad_language.py](#)

5.8 Класс `bad_language.ToxicClassifier`

Открытые члены

- `def __init__(self, model_path='sismetanin/rubert-toxic-pikabu-2ch')`
- `def __call__(self, message)`
- `def is_toxic(self, message)`
- `def how_toxic(self, message)`

Открытые атрибуты

- [toxic_tokenizer](#)
- [toxic_bert](#)

5.8.1 Подробное описание

См. определение в файле `bad_language.py` строка 7

5.8.2 Конструктор(ы)

5.8.2.1 `__init__()`

```
def bad_language.ToxicClassifier.__init__(  
    self,  
    model_path = 'sismetanin/rubert-toxic-pikabu-2ch ' )
```

См. определение в файле `bad_language.py` строка 8

5.8.3 Методы

5.8.3.1 `__call__()`

```
def bad_language.ToxicClassifier.__call__(  
    self,  
    message )
```

См. определение в файле `bad_language.py` строка 12

5.8.3.2 `how_toxic()`

```
def bad_language.ToxicClassifier.how_toxic (  
    self,  
    message )
```

См. определение в файле `bad_language.py` строка 18

5.8.3.3 `is_toxic()`

```
def bad_language.ToxicClassifier.is_toxic (  
    self,  
    message )
```

См. определение в файле `bad_language.py` строка 15

5.8.4 Данные класса

5.8.4.1 `toxic_bert`

```
bad_language.ToxicClassifier.toxic_bert
```

См. определение в файле `bad_language.py` строка 10

5.8.4.2 `toxic_tokenizer`

```
bad_language.ToxicClassifier.toxic_tokenizer
```

См. определение в файле `bad_language.py` строка 9

Объявления и описания членов класса находятся в файле:

- [utils/bad_language.py](#)

5.9 Класс prod.worker.Worker

Класс для работы с Telegram API.

Открытые члены

- `def __init__ (self, token)`
Создаёт объект класса `Worker` и достаёт необходимые для работы файлы
- `def work_once (self)`
Получает текущее состояние чат-бота и обрабатывает одно сообщение
- `def work (self)`
Основной цикл работы программы

Открытые атрибуты

- `chat_bots`
Словарь из id чата в чат-бот по работе с этим чатом
- `offset`
Текущее отклонение от сообщений Необходимо, чтобы старые сообщения не обрабатывались
- `token`
Токен для работы с Telegram API.

5.9.1 Подробное описание

Класс для работы с Telegram API.

См. определение в файле worker.py строка 13

5.9.2 Конструктор(ы)

5.9.2.1 __init__()

```
def prod.worker.Worker.__init__ (
    self,
    token )
```

Создаёт объект класса `Worker` и достаёт необходимые для работы файлы

Аргументы

token	Токен для работы с Telegram API
-------	---------------------------------

См. определение в файле worker.py строка 17

5.9.3 Методы

5.9.3.1 `work()`

```
def prod.worker.Worker.work (
    self )
```

Основной цикл работы программы

См. определение в файле `worker.py` строка 64

5.9.3.2 `work_once()`

```
def prod.worker.Worker.work_once (
    self )
```

Получает текущее состояние чат-бота и обрабатывает одно сообщение

См. определение в файле `worker.py` строка 38

5.9.4 Данные класса

5.9.4.1 `chat_bots`

```
prod.worker.Worker.chat_bots
```

Словарь из `id` чата в чат-бот по работе с этим чатом

См. определение в файле `worker.py` строка 20

5.9.4.2 `offset`

```
prod.worker.Worker.offset
```

Текущее отклонение от сообщений Необходимо, чтобы старые сообщения не обрабатывались

См. определение в файле `worker.py` строка 25

5.9.4.3 `token`

```
prod.worker.Worker.token
```

Токен для работы с Telegram API.

См. определение в файле `worker.py` строка 29

Объявления и описания членов класса находятся в файле:

- [worker.py](#)

Глава 6

Файлы

6.1 Файл `__init__.py`

Пространства имен

- `prod`
Содержит всё для работы Telegram бота

Переменные

- `prod.parser = argparse.ArgumentParser()`
- `prod.required`
- `prod.True`
- `prod.help`
- `prod.args = parser.parse_args()`

6.2 Файл `utils/bad_language.py`

Классы

- `class bad_language.ToxicClassifier`
- `class bad_language.SwearDetector`
- `class bad_language.Merger`

Пространства имен

- `bad_language`

6.3 Файл `utils/embeddings.py`

Классы

- `class embeddings.EmbeddingCreator`
- `class embeddings.QuestionFinder`

Пространства имен

- [embeddings](#)

6.4 Файл `utils/JokeClassifier.py`

Классы

- class [JokeClassifier.JokeClassifier](#)

Пространства имен

- [JokeClassifier](#)

6.5 Файл `utils/telegram/dialogpt.py`

Классы

- class [dialogpt.DialoGPT](#)
Класс, содержащий в себе интерфейс работы с [DialoGPT](#).

Пространства имен

- [dialogpt](#)
Содержит класс чат-бота [DialoGPT](#).

6.6 Файл `utils/telegram/text.py`

Пространства имен

- [text](#)
Содержит функции для работы с текстами и с Telegram.

Функции

- def [text.extract_text](#) (message)
Извлекает текст из сообщения
- def [text.get_text](#) (message)
Получает и обрабатывает текст из сообщения
- def [text.get_joke](#) ()
Выводит случайный анекдот из набора случайных анекдотов

6.7 Файл `utils/telegram/translate_emoji.py`

Пространства имен

- `translate_emoji`
Содержит материалы для обработки эмоджи

Функции

- `def translate_emoji.eng_to_rus (text)`
Переводит текст с английского на русский
- `def translate_emoji.emoji_to_text (emoji)`
Переводит эмоджи в текст на русском
- `def translate_emoji.de_emojify (text)`
Переводит эмоджи в тексте на русский язык

Переменные

- `translate_emoji.translator = google_translator()`
Переводчик

6.8 Файл `utils/telegram/web.py`

Пространства имен

- `web`
Содержит функции для взаимодействия с веб-сервисами

Функции

- `def web.get_updates (token, offset)`
Получает новые сообщения с сервера.
- `def web.send_message (token, chat_id, text)`
Отправляет сообщение в чат.
- `def web.get_advice ()`
Достаёт случайный совет

Переменные

- `string web.api_url = 'https://api.telegram.org/'`
Ссылка на Telegram API.

6.9 Файл `worker.py`

Классы

- `class prod.worker.Worker`
Класс для работы с Telegram API.

Пространства имен

- `prod.worker`
Содержит класс `Worker`.

Предметный указатель

- `-- call --`
 - `bad_language.Merger`, [27](#)
 - `bad_language.SwearDetector`, [30](#)
 - `bad_language.ToxicClassifier`, [32](#)
 - `JokeClassifier.JokeClassifier`, [26](#)
 - `-- init --`
 - `bad_language.Merger`, [27](#)
 - `bad_language.SwearDetector`, [30](#)
 - `bad_language.ToxicClassifier`, [32](#)
 - `dialogpt.DialoGPT`, [18](#)
 - `embeddings.EmbeddingCreator`, [22](#)
 - `embeddings.QuestionFinder`, [28](#)
 - `JokeClassifier.JokeClassifier`, [25](#)
 - `prod.worker.Worker`, [34](#)
- `-- init --.py`, [37](#)
- `api_url`
 - `web`, [15](#)
- `args`
 - `prod`, [8](#)
- `bad_language`, [7](#)
- `bad_language.Merger`, [27](#)
 - `-- call --`, [27](#)
 - `-- init --`, [27](#)
 - `classifiers`, [27](#)
 - `is_soft`, [28](#)
- `bad_language.SwearDetector`, [30](#)
 - `-- call --`, [30](#)
 - `-- init --`, [30](#)
 - `blocklist`, [31](#)
 - `clear`, [31](#)
 - `find_swear`, [31](#)
 - `has_swear`, [31](#)
 - `lemmatizer`, [31](#)
- `bad_language.ToxicClassifier`, [32](#)
 - `-- call --`, [32](#)
 - `-- init --`, [32](#)
 - `how_toxic`, [33](#)
 - `is_toxic`, [33](#)
 - `toxic_bert`, [33](#)
 - `toxic_tokenizer`, [33](#)
- `blocklist`
 - `bad_language.SwearDetector`, [31](#)
- `chat_bots`
 - `prod.worker.Worker`, [35](#)
- `chat_history`
 - `dialogpt.DialoGPT`, [20](#)
- `classifiers`
 - `bad_language.Merger`, [27](#)
- `clear`
 - `bad_language.SwearDetector`, [31](#)
- `de_emojify`
 - `translate_emoji`, [11](#)
- `device`
 - `embeddings.EmbeddingCreator`, [23](#)
- `dialogpt`, [7](#)
- `dialogpt.DialoGPT`, [17](#)
 - `-- init --`, [18](#)
 - `chat_history`, [20](#)
 - `generate`, [18](#)
 - `get_length_param`, [19](#)
 - `get_response`, [19](#)
 - `model`, [20](#)
 - `process_text`, [19](#)
 - `restart`, [20](#)
 - `tokenizer`, [20](#)
 - `user_input_size`, [21](#)
 - `window_size`, [21](#)
- `dim`
 - `embeddings.QuestionFinder`, [29](#)
- `embeddings`, [7](#)
 - `embeddings.EmbeddingCreator`, [23](#)
 - `embeddings.QuestionFinder`, [29](#)
- `embeddings.EmbeddingCreator`, [21](#)
 - `-- init --`, [22](#)
 - `device`, [23](#)
 - `embeddings`, [23](#)
 - `get_by_index`, [22](#)
 - `get_phrase_embedding`, [22](#)
 - `make_embeddings`, [22](#)
 - `model`, [24](#)
 - `run`, [23](#)
 - `tokenize`, [23](#)
 - `tokenized`, [24](#)
 - `tokenizer`, [24](#)
 - `words`, [24](#)
 - `write_matrix`, [23](#)
- `embeddings.QuestionFinder`, [28](#)
 - `-- init --`, [28](#)
 - `dim`, [29](#)
 - `embeddings`, [29](#)
 - `find_closest`, [29](#)
 - `labels`, [29](#)
 - `num`, [29](#)
 - `p`, [29](#)

- emoji_to_text
 - translate_emoji, 12
- eng_to_rus
 - translate_emoji, 12
- extract_text
 - text, 10
- find_closest
 - embeddings.QuestionFinder, 29
- find_swear
 - bad_language.SwearDetector, 31
- generate
 - dialogpt.DialoGPT, 18
- get_advice
 - web, 13
- get_by_index
 - embeddings.EmbeddingCreator, 22
- get_joke
 - text, 10
- get_length_param
 - dialogpt.DialoGPT, 19
- get_phrase_embedding
 - embeddings.EmbeddingCreator, 22
- get_response
 - dialogpt.DialoGPT, 19
- get_text
 - text, 10
- get_updates
 - web, 14
- has_swear
 - bad_language.SwearDetector, 31
- help
 - prod, 8
- how_toxic
 - bad_language.ToxicClassifier, 33
- is_soft
 - bad_language.Merger, 28
- is_toxic
 - bad_language.ToxicClassifier, 33
- JokeClassifier, 8, 24
 - JokeClassifier.JokeClassifier, 25
 - __call__, 26
 - __init__, 25
 - model, 26
 - tokenizer, 26
- labels
 - embeddings.QuestionFinder, 29
- lemmatizer
 - bad_language.SwearDetector, 31
- make_embeddings
 - embeddings.EmbeddingCreator, 22
- model
 - dialogpt.DialoGPT, 20
 - embeddings.EmbeddingCreator, 24
- JokeClassifier.JokeClassifier, 26
- num
 - embeddings.QuestionFinder, 29
- offset
 - prod.worker.Worker, 35
- p
 - embeddings.QuestionFinder, 29
- parser
 - prod, 8
- process_text
 - dialogpt.DialoGPT, 19
- prod, 8
 - args, 8
 - help, 8
 - parser, 8
 - required, 9
 - True, 9
- prod.worker, 9
 - prod.worker.Worker, 34
 - __init__, 34
 - chat_bots, 35
 - offset, 35
 - token, 35
 - work, 35
 - work_once, 35
- required
 - prod, 9
- restart
 - dialogpt.DialoGPT, 20
- run
 - embeddings.EmbeddingCreator, 23
- send_message
 - web, 14
- text, 9
 - extract_text, 10
 - get_joke, 10
 - get_text, 10
- token
 - prod.worker.Worker, 35
- tokenize
 - embeddings.EmbeddingCreator, 23
- tokenized
 - embeddings.EmbeddingCreator, 24
- tokenizer
 - dialogpt.DialoGPT, 20
 - embeddings.EmbeddingCreator, 24
 - JokeClassifier.JokeClassifier, 26
- toxic_bert
 - bad_language.ToxicClassifier, 33
- toxic_tokenizer
 - bad_language.ToxicClassifier, 33
- translate_emoji, 11
 - de_emojify, 11
 - emoji_to_text, 12

- eng_to_rus, [12](#)
 - translator, [13](#)
- translator
 - translate_emoji, [13](#)
- True
 - prod, [9](#)
- user_input_size
 - dialogpt.DialoGPT, [21](#)
- utils/bad_language.py, [37](#)
- utils/embeddings.py, [37](#)
- utils/JokeClassifier.py, [38](#)
- utils/telegram/dialogpt.py, [38](#)
- utils/telegram/text.py, [38](#)
- utils/telegram/translate_emoji.py, [39](#)
- utils/telegram/web.py, [39](#)
- web, [13](#)
 - api_url, [15](#)
 - get_advice, [13](#)
 - get_updates, [14](#)
 - send_message, [14](#)
- window_size
 - dialogpt.DialoGPT, [21](#)
- words
 - embeddings.EmbeddingCreator, [24](#)
- work
 - prod.worker.Worker, [35](#)
- work_once
 - prod.worker.Worker, [35](#)
- worker.py, [39](#)
- write_matrix
 - embeddings.EmbeddingCreator, [23](#)