
metamorph

APN-Pucky

Dec 05, 2022

CONTENTS:

1	Install	1
1.1	metamorph	1
2	Indices and tables	7
	Python Module Index	9
	Index	11

INSTALL

Stable version:

```
$ pip install metamorph
```

Dev version:

```
$ pip install --index-url https://test.pypi.org/simple/ metamorph
```

1.1 metamorph

1.1.1 Submodules

`metamorph.config`

Module Contents

Classes

<i>Config</i>	Defines the language flow to generate alternative texts.
---------------	--

Functions

<i>is_end</i> (dic)	Returns True if dictionary is an end node.
<i>no_extra</i> (dic)	Returns keys of dictionary that are not 'extra'.
<i>remove_lower</i> (text)	Remove lowercase letters from <code>text</code> .

`metamorph.config.is_end(dic)`

Returns True if dictionary is an end node.

`metamorph.config.no_extra(dic)`

Returns keys of dictionary that are not 'extra'.

`metamorph.config.remove_lower(text: str)`

Remove lowercase letters from `text`.

Parameters

text – Text to remove lowercase letters from.

Returns

Text without lowercase letters.

Example::

```
>>> remove_lower("GoogleTranslate")
'GT'
```

```
class metamorph.config.Config(file: str = None, start='en', goal='en', translator='GoogleTranslator',
                               flow=None)
```

Defines the language flow to generate alternative texts.

load_file(file: str)

Loads a configuration file.

default_extra(direct, k)

Adds default keys to dictionary at `direct[k]`.

fill_missing(direct)

Sets default extras for missing elements in dictionary.

str_diagram(nodes='language', arrows=None)

Prints a diagram of the language flow.

_recursive_get_str_max_length(sub, key)

_recursive_str_diagram(sub, kk, depth=1, lines=None, nodes='language', arrows=None, len_nodes=None, len_arrows=None)

metamorph.handler**Module Contents****Functions**

<code>generate_alternatives</code> (text, conf)	Generate alternatives for <code>text</code> using Config <code>conf</code> .
<code>recursive_translate</code> (conf, sub, kk)	Recursively translate <code>sub</code> using Config <code>conf</code> .
<code>translate</code> (translator, source, target, text[, quiet, ...])	Translate <code>text</code> from <code>source</code> language to <code>target</code> language using translator <code>translator</code> .

`metamorph.handler.generate_alternatives`(text, conf)

Generate alternatives for `text` using Config `conf`.

Parameters

- **text** – Text to generate alternatives for.
- **conf** – Config to use.

Returns

List of alternatives.

Example::

```
>>> from metamorph.config import Config
>>> "Hello World!" in generate_alternatives("Hallo world!", Config(flow={"de":None, "fr":None, "es":None}))
True
>>> "Hallo Welt!" in generate_alternatives("Hello world!", Config(start="de", goal="de", flow={"en":None, "fr":None, "es":None}))
True
>>> "Hello World!" in generate_alternatives("Hallo world!", Config("default_config.yaml"))
True
```

`metamorph.handler.recursive_translate(conf, sub, kk)`

Recursively translate sub using Config conf.

`metamorph.handler.translate(translator, source, target, text, quiet=False, verbose=True)`

Translate text from source language to target language using translator translator. :param translator: Translator to use (from deep_translator).

Parameters

- **source** – Source language.
- **target** – Target language.
- **text** – Text to translate.
- **quiet** – If True, don't print anything.
- **verbose** – If True, print error messages.

Returns

Translated text.

Example::

```
>>> translate(GoogleTranslator, "en", "de", "Hello world!")
'Hallo Welt!'
```

`metamorph.main`

Module Contents**Functions**

<code>__main__()</code>	Main function.
-------------------------	----------------

Attributes

get_edits_string

metamorph.main.get_edits_string

metamorph.main.__main__()

Main function.

metamorph.util

Module Contents

Functions

get_edits_string(old, new)

Colorize the differences between two strings.

metamorph.util.get_edits_string(*old*, *new*)

Colorize the differences between two strings.

1.1.2 Package Contents

Classes

Config

Defines the language flow to generate alternative texts.

Functions

generate_alternatives(text, conf)

Generate alternatives for `text` using Config `conf`.

translate(translator, source, target, text[, quiet, ...])

Translate `text` from `source` language to `target` language using translator `translator`.

metamorph.generate_alternatives(*text*, *conf*)

Generate alternatives for `text` using Config `conf`.

Parameters

- **text** – Text to generate alternatives for.
- **conf** – Config to use.

Returns

List of alternatives.

Example::


```

>>> from metamorph.config import Config
>>> "Hello World!" in generate_alternatives("Hallo world!", Config(flow={"de":None, "fr":None, "es":None}))
True
>>> "Hallo Welt!" in generate_alternatives("Hello world!", Config(start="de", goal="de", flow={"en":None, "fr":None, "es":None}))
True
>>> "Hello World!" in generate_alternatives("Hallo world!", Config("default_config.yaml"))
True

```

metamorph.translate(*translator, source, target, text, quiet=False, verbose=True*)

Translate text from source language to target language using translator *translator*. :param *translator*: Translator to use (from *deep_translator*).

Parameters

- **source** – Source language.
- **target** – Target language.
- **text** – Text to translate.
- **quiet** – If True, don't print anything.
- **verbose** – If True, print error messages.

Returns

Translated text.

Example::

```

>>> translate(GoogleTranslator, "en", "de", "Hello world!")
'Hallo Welt!'

```

class metamorph.Config(*file: str = None, start='en', goal='en', translator='GoogleTranslator', flow=None*)

Defines the language flow to generate alternative texts.

load_file(*file: str*)

Loads a configuration file.

default_extra(*direct, k*)

Adds default keys to dictionary at *direct[k]*.

fill_missing(*direct*)

Sets default extras for missing elements in dictionary.

str_diagram(*nodes='language', arrows=None*)

Prints a diagram of the language flow.

_recursive_get_str_max_length(*sub, key*)

_recursive_str_diagram(*sub, kk, depth=1, lines=None, nodes='language', arrows=None, len_nodes=None, len_arrows=None*)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

m

- `metamorph`, 1
- `metamorph.config`, 1
- `metamorph.handler`, 2
- `metamorph.main`, 3
- `metamorph.util`, 4

Symbols

`__main__()` (in module `metamorph.main`), 4
`_recursive_get_str_max_length()` (metamorph.Config method), 5
`_recursive_get_str_max_length()` (metamorph.config.Config method), 2
`_recursive_str_diagram()` (metamorph.Config method), 5
`_recursive_str_diagram()` (metamorph.config.Config method), 2

C

`Config` (class in `metamorph`), 5
`Config` (class in `metamorph.config`), 2

D

`default_extra()` (metamorph.Config method), 5
`default_extra()` (metamorph.config.Config method), 2

F

`fill_missing()` (metamorph.Config method), 5
`fill_missing()` (metamorph.config.Config method), 2

G

`generate_alternatives()` (in module `metamorph`), 4
`generate_alternatives()` (in module `metamorph.handler`), 2
`get_edits_string` (in module `metamorph.main`), 4
`get_edits_string()` (in module `metamorph.util`), 4

I

`is_end()` (in module `metamorph.config`), 1

L

`load_file()` (metamorph.Config method), 5
`load_file()` (metamorph.config.Config method), 2

M

`metamorph`
 module, 1
`metamorph.config`

 module, 1
`metamorph.handler`
 module, 2
`metamorph.main`
 module, 3
`metamorph.util`
 module, 4
`module`
 metamorph, 1
 metamorph.config, 1
 metamorph.handler, 2
 metamorph.main, 3
 metamorph.util, 4

N

`no_extra()` (in module `metamorph.config`), 1

R

`recursive_translate()` (in module `metamorph.handler`), 3
`remove_lower()` (in module `metamorph.config`), 1

S

`str_diagram()` (metamorph.Config method), 5
`str_diagram()` (metamorph.config.Config method), 2

T

`translate()` (in module `metamorph`), 5
`translate()` (in module `metamorph.handler`), 3