metamorph

APN-Pucky

Dec 05, 2022

CONTENTS:

CHAPTER

ONE

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

Config	Defines the language flow to generate alternative texts.

Functions

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

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

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

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

__main__()	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*)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

Symbols

C

D

F

G

I

L

M

N

R

S

T