metamorph

Alexander Puck Neuwirth <alexander@neuwirth-informatik.de>

Feb 21, 2024

CONTENTS:

1 metamorph **1**
- 1.1 Documentation . . . 1
- 1.2 Versions . . . 1
- 1.3 Configuration . . . 2

2 API Reference **3**
- 2.1 `metamorph` . . . 3

3 Indices and tables **9**

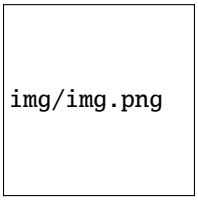
Python Module Index **11**

Index **13**

CHAPTER

ONE

METAMORPH



First line is the input followed by colorized suggestions.

Metamorph easily fixes typos and suggests alternative wordings by repeatedly translating the text into different languages and in the end to the desired language.

Doc: `metamorph --help`

Stable	Unstable

1.1 Documentation

- https://metamorph-apn.readthedocs.io/en/stable/
- https://apn-pucky.github.io/metamorph/index.html

1.2 Versions

1.2.1 Stable

```
pip install metamorph [--user] [--upgrade]
```

1.2.2 Dev

```
pip install --index-url https://test.pypi.org/simple/ metamorph [--user] [--upgrade]
```

1.3 Configuration

For a list of parameters run `metamorph -h`.

The root node `flow` can have multiple different starting languages (given `start` is None).

```yaml
translator: "GoogleTranslator"
start: "de"
goal: "de"

flow:
  de:
    fr:
      es:
        fr:
    de:
      es:
      fr:
        sv:
  fr:
    en:
  en:
  fi:
    de:
      fr:
        es:
          fr:
      de:
        es:
        fr:
          sv:
  sv:
```

This exemplary `configs/config.yaml` will produce following results (note `-sd` for diagrams and `-c` for config, while most command line parameters take precedence over config (`-gs` here)). A list of translators can be found here https://github.com/nidhaloff/deep-translator.

```
metamorph -i -sd -gs en -c config.yaml
```

(`GoogleTranslate` gets abbreviated to only capital letters `GT`)

CHAPTER

TWO

API REFERENCE

This page contains auto-generated API reference documentation[1].

2.1 metamorph

2.1.1 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.

[1] Created with sphinx-autoapi

Example::

```
>>> remove_lower("GoogleTranslate")
'GT'
>>> remove_lower({"GoogleTranslate":"XXX"})
{'GT': 'XXX'}
```

class metamorph.config.**Config**(*file: str = None*, *start='en'*, *goal='en'*, *translator='GoogleTranslator'*, *proxies=None*, *api_keys=None*, *flow=None*, *color='green'*, *on_color='on_red'*)

Defines the language flow to generate alternative texts.

get_api_key(*translator*)

Returns the api key for `translator`.

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[, api_key, ...])	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("configs/
↪default_config.yaml"))
True
```

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

Recursively translate sub using Config conf.

metamorph.handler.**translate**(*translator*, *source*, *target*, *text*, *api_key=None*, *proxies=None*, *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[, color, on_color])	Colorize the differences between two strings.

metamorph.util.**get_edits_string**(*old: str*, *new: str*, *color: str = 'green'*, *on_color: str = 'on_red'*)

Colorize the differences between two strings.

2.1.2 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[, api_key, ...])	Translate `text` from `source` language to `target` language using translator `translator`.

class metamorph.**Config**(*file: str = None*, *start='en'*, *goal='en'*, *translator='GoogleTranslator'*, *proxies=None*, *api_keys=None*, *flow=None*, *color='green'*, *on_color='on_red'*)

Defines the language flow to generate alternative texts.

get_api_key(*translator*)

Returns the api key for `translator`.

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.**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("configs/
↪default_config.yaml"))
True
```

metamorph.**translate**(*translator*, *source*, *target*, *text*, *api_key=None*, *proxies=None*, *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!'
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

metamorph, 3
metamorph.config, 3
metamorph.handler, 4
metamorph.main, 5
metamorph.util, 6

INDEX

Symbols

__main__() (*in module metamorph.main*), 6
_recursive_get_str_max_length() (*metamorph.Config method*), 7
_recursive_get_str_max_length() (*metamorph.config.Config method*), 4
_recursive_str_diagram() (*metamorph.Config method*), 7
_recursive_str_diagram() (*metamorph.config.Config method*), 4

C

Config (*class in metamorph*), 6
Config (*class in metamorph.config*), 4

D

default_extra() (*metamorph.Config method*), 6
default_extra() (*metamorph.config.Config method*), 4

F

fill_missing() (*metamorph.Config method*), 7
fill_missing() (*metamorph.config.Config method*), 4

G

generate_alternatives() (*in module metamorph*), 7
generate_alternatives() (*in module metamorph.handler*), 4
get_api_key() (*metamorph.Config method*), 6
get_api_key() (*metamorph.config.Config method*), 4
get_edits_string (*in module metamorph.main*), 6
get_edits_string() (*in module metamorph.util*), 6

I

is_end() (*in module metamorph.config*), 3

L

load_file() (*metamorph.Config method*), 6
load_file() (*metamorph.config.Config method*), 4

M

metamorph
    module, 3
metamorph.config
    module, 3
metamorph.handler
    module, 4
metamorph.main
    module, 5
metamorph.util
    module, 6
module
    metamorph, 3
    metamorph.config, 3
    metamorph.handler, 4
    metamorph.main, 5
    metamorph.util, 6

N

no_extra() (*in module metamorph.config*), 3

R

recursive_translate() (*in module metamorph.handler*), 5
remove_lower() (*in module metamorph.config*), 3

S

str_diagram() (*metamorph.Config method*), 7
str_diagram() (*metamorph.config.Config method*), 4

T

translate() (*in module metamorph*), 7
translate() (*in module metamorph.handler*), 5