Package 'elbird'

August 12, 2022

Title Blazing Fast Morphological Analyzer Based on Kiwi(Korean Intelligent Word Identifier)

Version 0.2.5

Description This is the R wrapper package Kiwi(Korean Intelligent Word Identifier), a blazing fast speed morphological analyzer for Korean. It supports configuration of user dictionary and detection of unregistered nouns based on frequency.

License LGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.1

URL https://github.com/mrchypark/elbird/

BugReports https://github.com/mrchypark/elbird/issues

SystemRequirements A valid Kiwi installation. Dynamic library is downloaded from <https://github.com/bab2min/Kiwi/releases> during the build step. See <https://github.com/bab2min/Kiwi> as well as for API documentation. A recent-enough compiler with C++11 support is required. git, curl and cmake required to build from source for kiwi.

Depends R (>= 3.5)

Imports dplyr, purrr, tibble, R6 (>= 2.4.0), vroom, matchr

LinkingTo cpp11

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Chanyub Park [aut, cre] (<https://orcid.org/0000-0001-6474-2570>)

Maintainer Chanyub Park <mrchypark@gmail.com>

Repository CRAN

Date/Publication 2022-08-12 15:50:02 UTC

R topics documented:

`analyze` *Simple version of analyze function.*

Description

Simple version of analyze function.

Usage

```
analyze(text, top_n = 3, match_option = Match$ALL, stopwords = FALSE)
```

Arguments

`text`	target text.
`top_n`	`integer`: Number of result. Default is 3.
`match_option`	`Match`: use Match. Default is Match$ALL
`stopwords`	stopwords option. Default is TRUE which is to use embaded stopwords dictionany. If FALSE, use not embaded stopwords dictionany. If char: path of dictionary txt file, use file. If `Stopwords` class, use it. If not valid value, work same as FALSE. Check `analyze()` how to use stopwords param.

Examples

```
## Not run:
  analyze("Test text.")
  analyze("Please use Korean.", top_n = 1)
  analyze("Test text.", 1, Match$ALL_WITH_NORMALIZING)
  analyze("Test text.", stopwords = FALSE)
  analyze("Test text.", stopwords = TRUE)
  analyze("Test text.", stopwords = "user_dict.txt")
  analyze("Test text.", stopwords = Stopwords$new(TRUE))

## End(Not run)
```

get_model *Get kiwi language model file.*

Description

Get kiwi language model file.

Usage

```
get_model(size = "base", path = model_home(), clean = FALSE)
```

Arguments

size	"small", "base", "large" model. default is "base". Also "all" available.
path	path for model files. default is model_home().
clean	remove previous model files before get new.

Source

https://github.com/bab2min/Kiwi/releases

Examples

```
## Not run:
  get_model("small")

## End(Not run)
```

Kiwi *Kiwi Class*

Description

Kiwi class is provide method for korean mophological analyze result.

Methods

Public methods:

- Kiwi$print()
- Kiwi$new()
- Kiwi$add_user_word()
- Kiwi$add_pre_analyzed_words()
- Kiwi$add_rules()
- Kiwi$load_user_dictionarys()

- Kiwi$extract_words()
- Kiwi$analyze()
- Kiwi$tokenize()
- Kiwi$split_into_sents()
- Kiwi$get_tidytext_func()
- Kiwi$clone()

Method `print()`**:** print method for `Kiwi` objects

Usage:

```
Kiwi$print(x, ...)
```

Arguments:

`x` self

`...` ignored

Method `new()`**:** Create a kiwi instance.

Usage:

```
Kiwi$new(
  num_workers = 0,
  model_size = "base",
  integrate_allomorph = TRUE,
  load_default_dict = TRUE
)
```

Arguments:

`num_workers int(optional)`: use multi-thread core number. default is 0 which means use all core.

`model_size char(optional)`: kiwi model select. default is "base". "small", "large" is available.

`integrate_allomorph bool(optional)`: default is TRUE.

`load_default_dict bool(optional)`: use defualt dictionary. default is TRUE.

Method `add_user_word()`**:** add user word with pos and score

Usage:

```
Kiwi$add_user_word(word, tag, score, orig_word = "")
```

Arguments:

`word char(required)`: target word to add.

`tag Tags(required)`: tag information about word.

`score num(required)`: score information about word.

`orig_word char(optional)`: origin word.

Method `add_pre_analyzed_words()`**:** TODO

Usage:

```
Kiwi$add_pre_analyzed_words(form, analyzed, score)
```

Arguments:

`form char(required)`: target word to add analyzed result.
`analyzed data.frame(required)`: analyzed result expected.
`score num(required)`: score information about pre analyzed result.

Method `add_rules()`**:** TODO

Usage:

```
Kiwi$add_rules(tag, pattern, replacement, score)
```

Arguments:

`tag Tags(required)`: target tag to add rules.
`pattern char(required)`: regular expression.
`replacement char(required)`: replace text.
`score num(required)`: score information about rules.

Method `load_user_dictionarys()`**:** add user dictionary using text file.

Usage:

```
Kiwi$load_user_dictionarys(user_dict_path)
```

Arguments:

`user_dict_path char(required)`: path of user dictionary file.

Method `extract_words()`**:** Extract Noun word candidate from texts.

Usage:

```
Kiwi$extract_words(
  input,
  min_cnt,
  max_word_len,
  min_score,
  pos_threshold,
  apply = FALSE
)
```

Arguments:

`input char(required)`: target text data
`min_cnt int(required)`: minimum count of word in text.
`max_word_len int(required)`: max word length.
`min_score num(required)`: minimum score.
`pos_threshold num(required)`: pos threashold.
`apply bool(optional)`: apply extracted word as user word dict.

Method `analyze()`**:** Analyze text to token and tag results.

Usage:

```
Kiwi$analyze(text, top_n = 3, match_option = Match$ALL, stopwords = FALSE)
```

Arguments:

`text char(required)`: target text.
`top_n int(optional)`: number of result. Default is 3.

`match_option` match_option `Match`: use Match. Default is Match$ALL

`stopwords` stopwords option. Default is FALSE which is use nothing. If `TRUE`, use embaded stopwords dictionany. If `char`: path of dictionary txt file, use file. If `Stopwords` class, use it. If not valid value, work same as FALSE.

Returns: `list` of result.

Method `tokenize()`**:** Analyze text to token and pos result just top 1.

Usage:

```
Kiwi$tokenize(
  text,
  match_option = Match$ALL,
  stopwords = FALSE,
  form = "tibble"
)
```

Arguments:

`text` `char(required)`: target text.

`match_option` match_option `Match`: use Match. Default is Match$ALL

`stopwords` stopwords option. Default is FALSE which is use nothing. If `TRUE`, use embaded stopwords dictionany. If `char`: path of dictionary txt file, use file. If `Stopwords` class, use it. If not valid value, work same as FALSE.

`form` `char(optional)`: return form. default is "tibble". "list", "tidytext" is available.

Method `split_into_sents()`**:** Some text may not split sentence by sentence. split_into_sents works split sentences to sentence by sentence.

Usage:

```
Kiwi$split_into_sents(text, match_option = Match$ALL, return_tokens = FALSE)
```

Arguments:

`text` `char(required)`: target text.

`match_option` match_option `Match`: use Match. Default is Match$ALL

`return_tokens` `bool(optional)`: add tokenized resault.

Method `get_tidytext_func()`**:** set function to tidytext unnest_tokens.

Usage:

```
Kiwi$get_tidytext_func(match_option = Match$ALL, stopwords = FALSE)
```

Arguments:

`match_option` match_option `Match`: use Match. Default is Match$ALL

`stopwords` stopwords option. Default is TRUE which is to use embaded stopwords dictionary. If FALSE, use not embaded stopwords dictionary. If char: path of dictionary txt file, use file. If `Stopwords` class, use it. If not valid value, work same as FALSE.

Returns: `function`

Examples:

```
\dontrun{
   kw <- Kiwi$new()
   tidytoken <- kw$get_tidytext_func()
   tidytoken("test")
}
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Kiwi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
  kw <- Kiwi$new()
  kw$analyze("test")
  kw$tokenize("test")

## End(Not run)

## ------------------------------------------------
## Method `Kiwi$get_tidytext_func`
## ------------------------------------------------

## Not run:
   kw <- Kiwi$new()
   tidytoken <- kw$get_tidytext_func()
   tidytoken("test")

## End(Not run)
```

`Match` *Analyze Match Options.*

Description

ALL option contains URL, EMAIL, HASHTAG, MENTION.

Usage

```
Match
```

Format

An object of class `EnumGenerator` of length 13.

Examples

```
## Not run:
 Match
 Match$ALL

## End(Not run)
```

`model_exists` *Verifies if model files exists.*

Description

Verifies if model files exists.

Usage

```
model_exists(size = "all")
```

Arguments

`size`	model size. default is "all" which is true that all three models must be present.

Value

`logical` model files exists or not.

Examples

```
## Not run:
  get_model("small")
  model_exists("small")

## End(Not run)
```

`model_home` *A simple exported version of* `kiwi_model_path()` *Returns the kiwi model path.*

Description

TODO explain ELBIRD_MODEL_HOME

Usage

```
model_home()
```

Value

character: file path

Examples

```
model_home()
```

---

model_works *Verifies if models work fine.*

Description

Verifies if models work fine.

Usage

```
model_works(size = "all")
```

Arguments

size	model size. default is "all" which is true that all three models must be present.

Value

logical model work or not.

Examples

```
## Not run:
  get_model("small")
  model_works("small")

## End(Not run)
```

---

split_into_sents *Split Sentences*

Description

Some text may not split sentence by sentence. split_into_sents works split sentences to sentence by sentence.

Usage

```
split_into_sents(text, return_tokens = FALSE)
```

Arguments

text	target text.
return_tokens	add tokenized resault.

Examples

```
## Not run:
 split_into_sents("text")
 split_into_sents("text", return_tokens = TRUE)

## End(Not run)
```

Stopwords *Stopwords Class*

Description

Stopwords is for filter result.

Methods

Public methods:

- Stopwords$print()
- Stopwords$new()
- Stopwords$add()
- Stopwords$add_from_dict()
- Stopwords$remove()
- Stopwords$save_dict()
- Stopwords$get()
- Stopwords$clone()

Method `print()`**:** print method for `Stopwords` objects

Usage:

```
Stopwords$print(x, ...)
```

Arguments:

`x` self

`...` ignored

Method `new()`**:** Create a stopwords object for filter stopwords on `analyze()` and `tokenize()` results.

Usage:

```
Stopwords$new(use_system_dict = TRUE)
```

Arguments:

`use_system_dict bool(optional)`: use system stopwords dictionary or not. Defualt is TRUE.

Method `add()`**:** add stopword one at a time.

Usage:

```
Stopwords$add(form = NA, tag = Tags$nnp)
```

Arguments:

`form char(optional)`: Form information. Default is NA.

`tag char(optional)`: Tag information. Default is "NNP". Please check Tags.

Examples:

```
\dontrun{
 sw <- Stopwords$new()
 sw$add("word", "NNG")
 sw$add("word", Tags$nng)
 }
```

Method `add_from_dict()`**:** add stopword from text file. text file need to form "TEXT/TAG". TEXT can remove like "/NNP". TAG required like "FORM/NNP".

Usage:

```
Stopwords$add_from_dict(path, dict_name = "user")
```

Arguments:

`path char(required)`: dictionary file path.

`dict_name char(optional)`: default is "user"

Method `remove()`**:** remove stopword one at a time.

Usage:

```
Stopwords$remove(form = NULL, tag = NULL)
```

Arguments:

`form char(optional)`: Form information. If form not set, remove tag in input.

`tag char(required)`: Tag information. Please check Tags.

Method `save_dict()`**:** save current stopwords list in text file.

Usage:

```
Stopwords$save_dict(path)
```

Arguments:

`path char(required)`: file path to save stopwords list.

Method `get()`**:** return tibble of stopwords.

Usage:

```
Stopwords$get()
```

Returns: a tibble for stopwords options for `analyze()` / `tokenize()` function.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Stopwords$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
  Stopwords$new()

## End(Not run)

## ------------------------------------------------
## Method `Stopwords$add`
## ------------------------------------------------

 ## Not run:
  sw <- Stopwords$new()
  sw$add("word", "NNG")
  sw$add("word", Tags$nng)

## End(Not run)
```

`Tags` *Tag list*

Description

Tags contains tag list for elbird.

Usage

```
Tags
```

Format

An object of class `EnumGenerator` of length 47.

Source

https://github.com/bab2min/Kiwi

Examples

```
 ## Not run:
  Tags
  Tags$nnp

## End(Not run)
```

tokenize *Simple version of tokenizer function.*

Description

Simple version of tokenizer function.

Usage

```
tokenize(text, match_option = Match$ALL, stopwords = TRUE)

tokenize_tbl(text, match_option = Match$ALL, stopwords = TRUE)

tokenize_tidytext(text, match_option = Match$ALL, stopwords = TRUE)

tokenize_tidy(text, match_option = Match$ALL, stopwords = TRUE)
```

Arguments

text	target text.
match_option	Match: use Match. Default is Match$ALL
stopwords	stopwords option. Default is TRUE which is to use embaded stopwords dictionany. If FALSE, use not embaded stopwords dictionany. If char: path of dictionary txt file, use file. If Stopwords class, use it. If not valid value, work same as FALSE. Check analyze() how to use stopwords param.

Value

list type of result.

Examples

```
## Not run:
  tokenize("Test text.")
  tokenize("Please use Korean.", Match$ALL_WITH_NORMALIZING)

## End(Not run)
```

Index