

Package ‘gibasa’

December 3, 2023

Type Package

Title An Alternative 'Rcpp' Wrapper of 'MeCab'

Version 1.0.1

Maintainer Akiru Kato <paithiov909@gmail.com>

Description A plain 'Rcpp' wrapper of 'MeCab' that can segment Chinese, Japanese, and Korean text into tokens. The main goal of this package is to provide an alternative to 'tidytext' using morphological analysis.

License GPL (>= 3)

URL <https://paithiov909.github.io/gibasa/>

BugReports <https://github.com/paithiov909/gibasa/issues>

Depends R (>= 2.10)

Imports dplyr, Matrix, purrr, Rcpp, RcppParallel, readr, rlang (>= 0.4.11), stringi, utils

Suggests roxygen2, testthat (>= 3.0.0), withr

LinkingTo Rcpp, RcppParallel

Config/Needs/website ggdendro, ggpubr, quanteda, quanteda.textmodels, quanteda.textplots, quanteda.textstats, reactable, stringr, textmineR, textrecipes, tidymodels, topicmodels, uwot, xgboost, paithiov909/ldccr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

SystemRequirements GNU make, MeCab

NeedsCompilation yes

Author Akiru Kato [aut, cre],
Shogo Ichinose [aut],
Taku Kudo [aut],
Jorge Nocedal [ctb],
Nippon Telegraph and Telephone Corporation [cph]

Repository CRAN

Date/Publication 2023-12-03 00:40:02 UTC

R topics documented:

as_tokens	2
bind_lr	3
bind_tf_idf2	4
build_sys_dic	5
build_user_dic	7
collapse_tokens	8
dictionary_info	9
gbs_tokenize	10
get_dict_features	11
ginga	11
is_blank	12
lex_density	13
mute_tokens	14
ngram_tokenizer	15
pack	15
prettify	16
tokenize	17

Index 19

as_tokens	<i>Create a list of tokens</i>
-----------	--------------------------------

Description

Create a list of tokens

Usage

```
as_tokens(
  tbl,
  token_field = "token",
  pos_field = get_dict_features()[1],
  nm = NULL
)
```

Arguments

tbl	A tibble of tokens out of tokenize().
token_field	<data-masked> Column name that contains tokens.
pos_field	Feature name that will be kept as the names of tokens. If you don't need them, give a NULL for this argument.
nm	Names of returned list. If left with NULL, "doc_id" field of tbl is used instead.

Value

A named list of tokens.

Examples

```
## Not run:
tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
) |>
  prettify(col_select = "POS1") |>
  as_tokens()

## End(Not run)
```

 bind_lr

Bind importance of bigrams

Description

Calculates and binds the importance of bigrams and their synergistic average.

Usage

```
bind_lr(tbl, term = "token", lr_mode = c("n", "dn"), avg_rate = 1)
```

Arguments

tbl	A tidy text dataset.
term	<data-masked> Column containing terms as string or symbol.
lr_mode	Method for computing 'FL' and 'FR' values. n is equivalent to 'LN' and 'RN', and dn is equivalent to 'LDN' and 'RDN'.
avg_rate	Weight of the 'LR' value.

Details

The 'LR' value is the synergistic average of bigram importance that based on the words and their positions (left or right side).

Value

A data.frame.

See Also

[doi:10.5715/jnlp.10.27](https://doi.org/10.5715/jnlp.10.27)

Examples

```
## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
)
bind_lr(df)

## End(Not run)
```

bind_tf_idf2

Bind term frequency and inverse document frequency

Description

Calculates and binds the term frequency, inverse document frequency, and TF-IDF of the dataset. This function experimentally supports 3 types of term frequencies and 4 types of inverse document frequencies, which are implemented in 'RMeCab' package.

Usage

```
bind_tf_idf2(
  tbl,
  term = "token",
  document = "doc_id",
  n = "n",
  tf = c("tf", "tf2", "tf3"),
  idf = c("idf", "idf2", "idf3", "idf4"),
  norm = FALSE,
  rmeCab_compat = TRUE
)
```

Arguments

tbl	A tidy text dataset.
term	<data-masked> Column containing terms as string or symbol.
document	<data-masked> Column containing document IDs as string or symbol.
n	<data-masked> Column containing document-term counts as string or symbol.
tf	Method for computing term frequency.
idf	Method for computing inverse document frequency.
norm	Logical; If passed as TRUE, the raw term counts are normalized being divided with L2 norms before computing IDF values.
rmeCab_compat	Logical; If passed as TRUE, computes values while taking care of compatibility with 'RMeCab'. Note that 'RMeCab' always computes IDF values using term frequency rather than raw term counts, and thus TF-IDF values may be doubly affected by term frequency.

Details

Types of term frequency can be switched with `tf` argument:

- `tf` is term frequency (not raw count of terms).
- `tf2` is logarithmic term frequency of which base is $\exp(1)$.
- `tf3` is binary-weighted term frequency.

Types of inverse document frequencies can be switched with `idf` argument:

- `idf` is inverse document frequency of which base is 2, with smoothed. 'smoothed' here means just adding 1 to raw counts after logarithmizing.
- `idf2` is global frequency IDF.
- `idf3` is probabilistic IDF of which base is 2.
- `idf4` is global entropy, not IDF in actual.

Value

A `data.frame`.

Examples

```
## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
) |>
dplyr::group_by(doc_id) |>
dplyr::count(token) |>
dplyr::ungroup()
bind_tf_idf2(df)

## End(Not run)
```

build_sys_dic

Build system dictionary

Description

Builds a UTF-8 system dictionary from source dictionary files.

Usage

```
build_sys_dic(dic_dir, out_dir, encoding)
```

Arguments

dic_dir	Directory where the source dictionaries are located. This argument is passed as '-d' option argument.
out_dir	Directory where the binary dictionary will be written. This argument is passed as '-o' option argument.
encoding	Encoding of input csv files. This argument is passed as '-f' option argument.

Details

This function is a wrapper around dictionary compiler of 'MeCab'.

Running this function will create 4 files: 'char.bin', 'matrix.bin', 'sys.dic', and 'unk.dic' in out_dir.

To use these compiled dictionary, you also need create a dicrc file in out_dir. A dicrc file is included in source dictionaries, so you can just copy it to out_dir.

Value

A TRUE is invisibly returned if dictionary is successfully built.

Examples

```
if (requireNamespace("withr")) {
  # create a sample dictionary in temporary directory
  build_sys_dic(
    dic_dir = system.file("latin", package = "gibasa"),
    out_dir = tempdir(),
    encoding = "utf8"
  )
  # copy the 'dicrc' file
  file.copy(
    system.file("latin/dicrc", package = "gibasa"),
    tempdir()
  )
  # mocking a 'mecabrc' file to temporarily use the dictionary
  withr::with_envvar(
    c(
      "MECABRC" = if (.Platform$OS.type == "windows") {
        "nul"
      } else {
        "/dev/null"
      },
      "RCPP_PARALLEL_BACKEND" = "tinythread"
    ),
    {
      tokenize("katta-wokattauresikatta", sys_dic = tempdir())
    }
  )
}
```

build_user_dic	<i>Build user dictionary</i>
----------------	------------------------------

Description

Builds a UTF-8 user dictionary from a csv file.

Usage

```
build_user_dic(dic_dir, file, csv_file, encoding)
```

Arguments

dic_dir	Directory where the source dictionaries are located. This argument is passed as '-d' option argument.
file	Path to write the user dictionary. This argument is passed as '-u' option argument.
csv_file	Path to an input csv file.
encoding	Encoding of input csv files. This argument is passed as '-f' option argument.

Details

This function is a wrapper around dictionary compiler of 'MeCab'.

Note that this function does not support auto assignment of word cost field. So, you can't leave any word costs as empty in your input csv file. To estimate word costs, use `posDebugRcpp()` function.

Value

A TRUE is invisibly returned if dictionary is successfully built.

Examples

```
if (requireNamespace("withr")) {  
  # create a sample dictionary in temporary directory  
  build_sys_dic(  
    dic_dir = system.file("latin", package = "gibasa"),  
    out_dir = tempdir(),  
    encoding = "utf8"  
  )  
  # copy the 'dicrc' file  
  file.copy(  
    system.file("latin/dicrc", package = "gibasa"),  
    tempdir()  
  )  
  # write a csv file and compile it into a user dictionary  
  csv_file <- tempfile(fileext = ".csv")  
  writeLines(  

```

```

    c(
      "qa, 0, 0, 5, \u304f\u3041",
      "qi, 0, 0, 5, \u304f\u3043",
      "qu, 0, 0, 5, \u304f",
      "qe, 0, 0, 5, \u304f\u3047",
      "qo, 0, 0, 5, \u304f\u3049"
    ),
    csv_file
  )
)
build_user_dic(
  dic_dir = tempdir(),
  file = (user_dic <- tempfile(fileext = ".dic")),
  csv_file = csv_file,
  encoding = "utf8"
)
# mocking a 'mecabrc' file to temporarily use the dictionary
withr::with_envvar(
  c(
    "MECABRC" = if (.Platform$OS.type == "windows") {
      "nul"
    } else {
      "/dev/null"
    },
    "RCPP_PARALLEL_BACKEND" = "tinythread"
  ),
  {
    tokenize("quensan", sys_dic = tempdir(), user_dic = user_dic)
  }
)
}

```

collapse_tokens

*Collapse sequences of tokens by condition***Description**

Concatenates sequences of tokens in the tidy text dataset, while grouping them by an expression.

Usage

```
collapse_tokens(tbl, condition, .collapse = "")
```

Arguments

tbl	A tidy text dataset.
condition	<data-masked> A logical expression.
.collapse	String with which tokens are concatenated.

Details

Note that this function drops all columns except but 'token' and columns for grouping sequences. So, the returned data.frame has only 'doc_id', 'sentence_id', 'token_id', and 'token' columns.

Value

A data.frame.

Examples

```
## Not run:
df <- tokenize(
  data.frame(
    doc_id = "odakyu-sen",
    text = "\u5c0f\u7530\u6025\u7dda"
  )
) |>
  prettify(col_select = "POS1")

head(collapse_tokens(
  df,
  POS1 == "\u540d\u8a5e" & stringr::str_detect(token, "^[\\p{Han}]+$")
))

## End(Not run)
```

dictionary_info

Get dictionary information

Description

Get dictionary information

Arguments

sys_dic String scalar.
user_dic String scalar.

Value

data.frame.

gbs_tokenize	<i>Tokenize sentences using 'MeCab'</i>
--------------	---

Description

Tokenize sentences using 'MeCab'

Usage

```
gbs_tokenize(
  x,
  sys_dic = "",
  user_dic = "",
  split = FALSE,
  partial = FALSE,
  mode = c("parse", "wakati")
)
```

Arguments

x	A data.frame like object or a character vector to be tokenized.
sys_dic	Character scalar; path to the system dictionary for 'MeCab'. Note that the system dictionary is expected to be compiled with UTF-8, not Shift-JIS or other encodings.
user_dic	Character scalar; path to the user dictionary for 'MeCab'.
split	Logical. When passed as TRUE, the function internally splits the sentences into sub-sentences using <code>stringi::stri_split_boundaries(type = "sentence")</code> .
partial	Logical. When passed as TRUE, activates partial parsing mode. To activate this feature, remember that all spaces at the start and end of the input chunks are already squashed. In particular, trailing spaces of chunks sometimes cause fatal errors.
mode	Character scalar to switch output format.

Value

A tibble or a named list of tokens.

get_dict_features	<i>Get dictionary features</i>
-------------------	--------------------------------

Description

Returns names of dictionary features. Currently supports "unidic17" (2.1.2 src schema), "unidic26" (2.1.2 bin schema), "unidic29" (schema used in 2.2.0, 2.3.0), "cc-cedict", "ko-dic" (mecab-ko-dic), "naist11", "sudachi", and "ipa".

Usage

```
get_dict_features(
    dict = c("ipa", "unidic17", "unidic26", "unidic29", "cc-cedict", "ko-dic", "naist11",
            "sudachi")
)
```

Arguments

dict	Character scalar; one of "ipa", "unidic17", "unidic26", "unidic29", "cc-cedict", "ko-dic", "naist11", or "sudachi".
------	---

Value

A character vector.

See Also

See also ['CC-CEDICT-MeCab'](#) and ['mecab-ko-dic'](#).

Examples

```
get_dict_features("ipa")
```

ginga	<i>Whole text of 'Ginga Tetsudo no Yoru' written by Miyazawa Kenji from Aozora Bunko</i>
-------	--

Description

Whole text of 'Ginga Tetsudo no Yoru' written by Miyazawa Kenji from Aozora Bunko

Usage

```
ginga
```

Format

An object of class character of length 553.

Details

A dataset containing the text of Miyazawa Kenji's novel "Ginga Tetsudo no Yoru" (English title: "Night on the Galactic Railroad") which was published in 1934, the year after Kenji's death. Copyright of this work has expired since more than 70 years have passed after the author's death.

The UTF-8 plain text is sourced from <https://www.aozora.gr.jp/cards/000081/card43737.html> and is cleaned of meta data.

Source

https://www.aozora.gr.jp/cards/000081/files/43737_ruby_19028.zip

Examples

```
head(ginga)
```

is_blank

Check if scalars are blank

Description

Check if scalars are blank

Usage

```
is_blank(x, trim = TRUE, ...)
```

Arguments

x	Object to check its emptiness.
trim	Logical.
...	Additional arguments for <code>base::sapply()</code> .

Value

Logical.

Examples

```
is_blank(list(c(a = "", b = NA_character_), NULL))
```

lex_density	<i>Calculate lexical density</i>
-------------	----------------------------------

Description

The lexical density is the proportion of content words (lexical items) in documents. This function is a simple helper for calculating the lexical density of given datasets.

Usage

```
lex_density(vec, contents_words, targets = NULL, negate = c(FALSE, FALSE))
```

Arguments

vec	A character vector.
contents_words	A character vector containing values to be counted as contents words.
targets	A character vector with which the denominator of lexical density is filtered before computing values.
negate	A logical vector of which length is 2. If passed as TRUE, then respectively negates the predicate functions for counting contents words or targets.

Value

A numeric vector.

Examples

```
## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
)
df |>
  prettify(col_select = "POS1") |>
  dplyr::group_by(doc_id) |>
  dplyr::summarise(
    noun_ratio = lex_density(POS1,
      "\u540d\u8a5e",
      c("\u52a9\u8a5e", "\u52a9\u52d5\u8a5e"),
      negate = c(FALSE, TRUE)
    ),
    mvr = lex_density(
      POS1,
      c("\u5f62\u5bb9\u8a5e", "\u526f\u8a5e", "\u9023\u4f53\u8a5e"),
      "\u52d5\u8a5e"
    ),
  ),
```

```

      vnr = lex_density(POS1, "\u52d5\u8a5e", "\u540d\u8a5e")
    )
## End(Not run)

```

 mute_tokens

Mute tokens by condition

Description

Permutes tokens in the tidy text dataset with a string scalar only if they are matched to an expression.

Usage

```
mute_tokens(tbl, condition, .as = NA_character_)
```

Arguments

tbl	A tidy text dataset.
condition	<data-masked> A logical expression.
.as	String with which tokens are replaced when they are matched to condition. The default value is NA_character.

Value

A data.frame.

Examples

```

## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
) |>
  prettify(col_select = "POS1")

head(mute_tokens(df, POS1 %in% c("\u52a9\u8a5e", "\u52a9\u52d5\u8a5e")))

## End(Not run)

```

ngram_tokenizer	<i>Ngrams tokenizer</i>
-----------------	-------------------------

Description

Make an ngram tokenizer function.

Usage

```
ngram_tokenizer(n = 1L)
```

Arguments

n Integer.

Value

ngram tokenizer function

Examples

```
bigram <- ngram_tokenizer(2)
bigram(letters, sep = "-")
```

pack	<i>Pack a data.frame of tokens</i>
------	------------------------------------

Description

Packs a data.frame of tokens into a new data.frame of corpus, which is compatible with the Text Interchange Formats.

Usage

```
pack(tbl, pull = "token", n = 1L, sep = "-", .collapse = " ")
```

Arguments

tbl A data.frame of tokens.

pull <[data-masked](#)> Column to be packed into text or ngrams body. Default value is token.

n Integer internally passed to ngrams tokenizer function created of `gibasa::ngram_tokenizer()`

sep Character scalar internally used as the concatenator of ngrams.

.collapse This argument is passed to `stringi::stri_c()`.

Value

A tibble.

Text Interchange Formats (TIF)

The Text Interchange Formats (TIF) is a set of standards that allows R text analysis packages to target defined inputs and outputs for corpora, tokens, and document-term matrices.

Valid data.frame of tokens

The data.frame of tokens here is a data.frame object compatible with the TIF.

A TIF valid data.frame of tokens are expected to have one unique key column (named doc_id) of each text and several feature columns of each tokens. The feature columns must contain at least token itself.

See Also

<https://github.com/ropenscilabs/tif>

Examples

```
## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
)
pack(df)

## End(Not run)
```

prettify

Prettify tokenized output

Description

Turns a single character column into features while separating with delimiter.

Usage

```
prettify(
  tbl,
  col = "feature",
  into = get_dict_features("ipa"),
  col_select = seq_along(into),
  delim = ",",
)
```

Arguments

tbl	A data.frame that has feature column to be prettified.
col	<data-masked> Column name where to be prettified.
into	Character vector that is used as column names of features.
col_select	Character or integer vector that will be kept in prettified features.
delim	Character scalar used to separate fields within a feature.

Value

A data.frame.

Examples

```
prettify(
  data.frame(x = c("x,y", "y,z", "z,x")),
  col = "x",
  into = c("a", "b"),
  col_select = "b"
)

## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
)
prettify(df, col_select = 1:3)
prettify(df, col_select = c(1, 3, 6))
prettify(df, col_select = c("POS1", "Original"))

## End(Not run)
```

tokenize

Tokenize sentences using 'MeCab'

Description

Tokenize sentences using 'MeCab'

Usage

```
tokenize(
  x,
  text_field = "text",
  docid_field = "doc_id",
  sys_dic = "",
```

```

user_dic = "",
split = FALSE,
partial = FALSE,
grain_size = 1L,
mode = c("parse", "wakati")
)

```

Arguments

<code>x</code>	A data.frame like object or a character vector to be tokenized.
<code>text_field</code>	<data-masked> String or symbol; column name where to get texts to be tokenized.
<code>docid_field</code>	<data-masked> String or symbol; column name where to get identifiers of texts.
<code>sys_dic</code>	Character scalar; path to the system dictionary for 'MeCab'. Note that the system dictionary is expected to be compiled with UTF-8, not Shift-JIS or other encodings.
<code>user_dic</code>	Character scalar; path to the user dictionary for 'MeCab'.
<code>split</code>	Logical. When passed as TRUE, the function internally splits the sentences into sub-sentences using <code>stringi::stri_split_boundaries(type = "sentence")</code> .
<code>partial</code>	Logical. When passed as TRUE, activates partial parsing mode. To activate this feature, remember that all spaces at the start and end of the input chunks are already squashed. In particular, trailing spaces of chunks sometimes cause fatal errors.
<code>grain_size</code>	Integer value larger than 1. This argument is internally passed to <code>RcppParallel::parallelFor</code> function. Setting a larger chunk size could improve the performance in some cases.
<code>mode</code>	Character scalar to switch output format.

Value

A tibble or a named list of tokens.

Examples

```

## Not run:
df <- tokenize(
  data.frame(
    doc_id = seq_along(ginga[5:8]),
    text = ginga[5:8]
  )
)
head(df)

## End(Not run)

```

Index

* datasets

ginga, [11](#)

as_tokens, [2](#)

bind_lr, [3](#)

bind_tf_idf2, [4](#)

build_sys_dic, [5](#)

build_user_dic, [7](#)

collapse_tokens, [8](#)

dictionary_info, [9](#)

gbs_tokenize, [10](#)

get_dict_features, [11](#)

ginga, [11](#)

is_blank, [12](#)

lex_density, [13](#)

mute_tokens, [14](#)

ngram_tokenizer, [15](#)

pack, [15](#)

prettify, [16](#)

tokenize, [17](#)