Package: vibrrt (via r-universe)

January 19, 2025

Type Package

Title An R Wrapper for 'Vibrato'

Version 0.1.0

Maintainer Akiru Kato <paithiov909@gmail.com>

Description An R wrapper for 'Vibrato', Viterbi-based accelerated tokenizer.

License MIT + file LICENSE

URL https://paithiov909.github.io/vibrrt/

BugReports https://github.com/paithiov909/vibrrt/issues

Depends R (>= 4.2)

Imports dplyr, Matrix, readr, rlang (>= 0.4.11), stringi

Suggests curl, hfhub, jsonlite, roxygen2, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements Cargo (rustc package manager)

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://paithiov909.r-universe.dev

RemoteUrl https://github.com/paithiov909/vibrrt

RemoteRef HEAD

RemoteSha 5c283ca162f8ccf030fe50785a3ef4fddc5dfd46

Contents

as_tokens *Create a list of tokens*

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 containing tokens.
pos_field	Column containing features 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.

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.
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

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 4 types of term frequencies and 5 types of inverse document frequencies.

Usage

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

Arguments

`tbl` A tidy text dataset.

`term` <`data-masked`> Column containing terms.

`document` <`data-masked`> Column containing document IDs.

`n` <`data-masked`> Column containing document-term counts.

`tf` Method for computing term frequency.

`idf` Method for computing inverse document frequency.

`norm` Logical; If passed as `TRUE`, TF-IDF values are normalized being divided with L2 norms.

`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.
- `itf` is inverse term frequency. Use with `idf="df"`.

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 values after logarithmizing.
- `idf2` is global frequency IDF.
- `idf3` is probabilistic IDF of which base is 2.
- `idf4` is global entropy, not IDF in actual.
- `df` is document frequency. Use with `tf="itf"`.

Value

A data.frame.

`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.

`get_dict_features` *Get dictionary features*

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", and "ipa".

Usage

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

Arguments

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

Value

A character vector.

See Also

See also 'CC-CEDICT-MeCab' and 'mecab-ko-dic'.

Examples

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

`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 `base::sapply()`.

Value

Logicals.

Examples

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

`lex_density` *Calculate lexical density*

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.

`mute_tokens` *Mute tokens by condition*

Description

Replaces 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.

ngram_tokenizer *Ngrams tokenizer*

Description

Makes 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 *Pack a data.frame of tokens*

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 `vibrrt::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 is 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

`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 containing features 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"
)
```

---

tokenize *Tokenize sentences using 'Vibrato'*

Description

Tokenize sentences using 'Vibrato'

Usage

```
tokenize(
  x,
  text_field = "text",
  docid_field = "doc_id",
  sys_dic = "",
  user_dic = "",
  split = FALSE,
  mode = c("parse", "wakati")
)
```

Arguments

`x`	A data.frame like object or a character vector to be tokenized.
`text_field`	<`data-masked`> String or symbol; column containing texts to be tokenized.
`docid_field`	<`data-masked`> String or symbol; column containing document IDs.
`sys_dic`	Character scalar; path to the system dictionary for 'Vibrato'.
`user_dic`	Character scalar; path to the user dictionary for 'Vibrato'.
`split`	split Logical. When passed as `TRUE`, the function internally splits the sentences into sub-sentences
`mode`	Character scalar to switch output format.

Value

A tibble or a named list of tokens.

Index