Package 'textclassificationtutorial'

August 26, 2026

Title Reproducible Text Classification Workflows

Version 0.1.2

Description Dependency-light tools and tutorials for teaching reproducible text classification. The package covers HTML text extraction, sentence segmentation, text preprocessing, document-term matrices, TF-IDF, keyword extraction, cosine similarity, stratified cross-validation, classification metrics, and a multinomial Naive Bayes classifier. It modernizes the code accompanying
Kobayashi, V. B., Berkers, H. A., Mol, S. T. Kismihok, G., and Den Hartog, D. N. (2017) <doi:10.1177/1094428117719322>
The package replaces the original scripts in the paper.

License Apache License (>= 2)

URL https://github.com/vkobayashi/textclassificationtutorial

BugReports https://github.com/vkobayashi/textclassificationtutorial/issues

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), xml2

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Vladimer Kobayashi [aut, cre],
Stefan Mol [aut],
Gabor Kismihok [aut]

Maintainer Vladimer Kobayashi <vladimer.kobayashi@gmail.com>

Repository CRAN

Date/Publication 2026-08-26 19:30:02 UTC

Contents

classification_metrics . . . 2
cosine_similarity . . . 3
document_term_matrix . . . 3
extract_html_dir . . . 4
extract_html_text . . . 5
extract_keywords . . . 6
fit_naive_bayes . . . 6
f_measure . . . 7
preprocess_text . . . 7
read_label_file . . . 8
split_sentences . . . 9
stratified_folds . . . 9
tf_idf . . . 10

Index **11**

`classification_metrics` *Calculate binary classification metrics*

Description

Calculate binary classification metrics

Usage

```
classification_metrics(truth, estimate, positive)
```

Arguments

`truth`	Vector of true classes.
`estimate`	Vector of predicted classes.
`positive`	Value identifying the positive class.

Value

A one-row data frame containing confusion counts, accuracy, balanced accuracy, precision, recall, specificity, and F1.

Examples

```
classification_metrics(c("task", "task", "other"), c("task", "other", "other"),
                       positive = "task")
```

cosine_similarity *Compute cosine similarity*

Description

Compute cosine similarity

Usage

```
cosine_similarity(x, y = NULL)
```

Arguments

x	Numeric matrix whose rows are observations.
y	Optional numeric matrix with the same columns as x. When omitted, computes all pairwise similarities among rows of x.

Value

A numeric similarity matrix. Similarities involving zero vectors are returned as NA.

Examples

```
x <- rbind(a = c(1, 1, 0), b = c(1, 0, 0), c = c(0, 0, 1))
cosine_similarity(x)
```

document_term_matrix *Construct a document-term matrix*

Description

Construct a document-term matrix

Usage

```
document_term_matrix(
  text,
  document_ids = NULL,
  binary = FALSE,
  min_doc_freq = 1L,
  max_doc_prop = 1
)
```

Arguments

`text`	Character vector containing one preprocessed document per item.
`document_ids`	Optional unique document identifiers.
`binary`	Logical; store term presence instead of term frequency?
`min_doc_freq`	Minimum number of documents in which a term must occur.
`max_doc_prop`	Maximum proportion of documents in which a term may occur.

Value

A numeric matrix with class `text_dtm`.

Examples

```
docs <- preprocess_text(c("data science", "data analysis", "science"))
document_term_matrix(docs)
```

`extract_html_dir` *Extract text from a directory of HTML files*

Description

Extract text from a directory of HTML files

Usage

```
extract_html_dir(path, pattern = "\\.html?$", recursive = FALSE, ...)
```

Arguments

`path`	Directory containing HTML files.
`pattern`	File-name regular expression.
`recursive`	Logical; search recursively?
`...`	Passed to `extract_html_text()`.

Value

A data frame with `document_id`, `path`, and `text`.

`extract_html_text` *Extract readable text from HTML*

Description

Extracts text from an HTML file or character string. When the suggested `xml2` package is installed, simple CSS selectors (`tag`, `.class`, `#id`, or `tag.class`) or XPath can target a specific part of the page. A dependency-free fallback strips markup from the full document.

Usage

```
extract_html_text(
  x,
  selector = NULL,
  xpath = NULL,
  collapse = "\n",
  trim = TRUE
)
```

Arguments

`x`	Path to an HTML file or a length-one HTML character string.
`selector`	Optional CSS selector.
`xpath`	Optional XPath expression. Supply at most one of `selector` and `xpath`.
`collapse`	Character used to join matched nodes.
`trim`	Logical; normalize whitespace and trim the result?

Value

A length-one character vector containing extracted text.

Examples

```
html <- "<html><body><h1>Analyst</h1><p>Analyze data.</p></body></html>"
extract_html_text(html)
```

extract_keywords *Extract top TF-IDF keywords*

Description

Extract top TF-IDF keywords

Usage

```
extract_keywords(x, n = 1L, already_tfidf = FALSE)
```

Arguments

x	A document-term matrix or TF-IDF matrix.
n	Number of keywords per document.
already_tfidf	Logical; is x already weighted?

Value

A data frame with document, rank, term, and weight.

fit_naive_bayes *Fit a multinomial Naive Bayes text classifier*

Description

Fit a multinomial Naive Bayes text classifier

Usage

```
fit_naive_bayes(x, y, laplace = 1, prior = NULL)
```

Arguments

x	Nonnegative numeric document-term matrix.
y	Class labels with one value per row of x.
laplace	Nonnegative additive smoothing parameter.
prior	Optional named class probabilities.

Value

An object of class text_nb.

Examples

```
x <- rbind(c(3, 0), c(2, 0), c(0, 3), c(0, 2))
colnames(x) <- c("analysis", "care")
model <- fit_naive_bayes(x, c("data", "data", "health", "health"))
predict(model, x)
```

`f_measure` *F-measure*

Description

F-measure

Usage

```
f_measure(precision, recall, beta = 1)
```

Arguments

`precision`	Numeric precision.
`recall`	Numeric recall.
`beta`	Relative weight assigned to recall.

Value

Numeric F-measure.

`preprocess_text` *Normalize text for document-term analysis*

Description

Normalize text for document-term analysis

Usage

```
preprocess_text(
  text,
  lowercase = TRUE,
  remove_punctuation = TRUE,
  remove_numbers = TRUE,
  stopwords = character(),
  min_token_length = 1L
)
```

Arguments

`text`	Character vector.
`lowercase`	Logical; convert text to lowercase?
`remove_punctuation`	Logical; replace punctuation with spaces?
`remove_numbers`	Logical; replace digits with spaces?
`stopwords`	Optional character vector of words to remove.
`min_token_length`	Minimum number of characters per token.

Value

A character vector of normalized documents.

Examples

```
preprocess_text(
  c("Analyze the data!", "Present 2 reports."),
  stopwords = c("the")
)
```

`read_label_file` *Read a one-label-per-line file*

Description

Read a one-label-per-line file

Usage

```
read_label_file(path, type = c("character", "integer", "numeric", "factor"))
```

Arguments

`path`	Path to a text file.
`type`	Return labels as character, integer, numeric, or factor.

Value

A vector of labels.

split_sentences *Split text into sentences*

Description

A lightweight sentence segmenter suitable for tutorials and clean prose. It splits at terminal punctuation followed by whitespace, and optionally at line breaks and vertical bars. For production multilingual segmentation, use a dedicated NLP tokenizer.

Usage

```
split_sentences(
  text,
  split_lines = TRUE,
  keep_punctuation = TRUE,
  drop_empty = TRUE
)
```

Arguments

`text`	Character vector.
`split_lines`	Logical; treat line breaks and `|` as boundaries?
`keep_punctuation`	Logical; retain terminal punctuation?
`drop_empty`	Logical; remove empty results?

Value

A character vector of sentences.

Examples

```
split_sentences("Analyze data. Present results! Work with teams?")
```

stratified_folds *Create stratified cross-validation folds*

Description

Create stratified cross-validation folds

Usage

```
stratified_folds(y, k = 5L, repeats = 1L, seed = NULL)
```

Arguments

y	Class labels.
k	Number of folds.
repeats	Number of repeated fold sets.
seed	Optional random seed. The caller's random-number state is restored.

Value

A list of integer test-set indices with class text_folds.

---

tf_idf *Calculate TF-IDF weights*

Description

Calculate TF-IDF weights

Usage

```
tf_idf(dtm, normalize = c("length", "max", "none"), smooth_idf = TRUE)
```

Arguments

dtm	Numeric document-term matrix.
normalize	Term-frequency normalization: document length, maximum frequency, or none.
smooth_idf	Logical; use smoothed inverse document frequency?

Value

A numeric matrix of TF-IDF weights.

Index

classification_metrics, 2
cosine_similarity, 3

document_term_matrix, 3

extract_html_dir, 4
extract_html_text, 5
extract_html_text(), *4*
extract_keywords, 6

f_measure, 7
fit_naive_bayes, 6

preprocess_text, 7

read_label_file, 8

split_sentences, 9
stratified_folds, 9

tf_idf, 10