Package ‘wordvector’

February 25, 2026

Type Package

Title Word and Document Vector Models

Version 0.6.1

Maintainer Kohei Watanabe <watanabe.kohei@gmail.com>

Description Create dense vector representation of words and documents using 'quanteda'. Implements Word2vec (Mikolov et al., 2013) <doi:10.48550/arXiv.1310.4546>, Doc2vec (Le & Mikolov, 2014) <doi:10.48550/ar
tent Semantic Analysis (Deerwester et al., 1990) <doi:10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9>.

URL https://github.com/koheiw/wordvector

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports quanteda (>= 4.1.0), methods, stringi, Matrix, proxyC, RSpectra, irlba, rsvd

Suggests testthat, word2vec, spelling

LinkingTo Rcpp, quanteda

Language en-US

LazyData true

NeedsCompilation yes

Author Kohei Watanabe [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6519-5265>),
Jan Wijffels [aut] (Original R code),
BNOSAC [cph] (Original R code),
Max Fomichev [ctb, cph] (Original C++ code)

Repository CRAN

Date/Publication 2026-02-25 11:50:02 UTC

Contents

`analogy` *Convert formula to named character vector*

Description

Convert a formula to a named character vector in analogy tasks.

Usage

```
analogy(formula)
```

Arguments

`formula`	a formula object that defines the relationship between words using + or - operators.

Value

a named character vector to be passed to `similarity()`.

See Also

`similarity()`

Examples

```
analogy(~ berlin - germany + france)
analogy(~ quick - quickly + slowly)
```

`as.matrix.textmodel_doc2vec` *Extract word or document vectors*

Description

Extract word or document vectors from a `textmodel_word2vec` or `textmodel_doc2vec` object.

Usage

```
## S3 method for class 'textmodel_doc2vec'
as.matrix(
  x,
  normalize = TRUE,
  layer = c("documents", "words"),
  group = FALSE,
  ...
)

## S3 method for class 'textmodel_word2vec'
as.matrix(x, normalize = TRUE, layer = "words", ...)
```

Arguments

`x`	a `textmodel_word2vec` or `textmodel_doc2vec` object.
`normalize`	if `TRUE`, returns normalized vectors.
`layer`	the layer from which the vectors are extracted.
`group`	[experimental] average sentence or paragraph vectors from the same document. Silently ignored when `layer = "words"`.
`...`	not used.

Value

a matrix that contain the word or document vectors in rows.

`as.textmodel_doc2vec` *Create distributed representation of documents*

Description

Create distributed representation of documents as weighted word vectors.

Usage

```
as.textmodel_doc2vec(
  x,
  model,
  normalize = FALSE,
  weights = 1,
  pattern = NULL,
  group_data = FALSE,
  ...
)
```

Arguments

`x`	a quanteda::tokens or quanteda::dfm object.
`model`	a textmodel_wordvector object.
`normalize`	if `TRUE`, normalized word vectors before creating document vectors.
`weights`	weight the word vectors by user-provided values; either a single value or multiple values sorted in the same order as the word vectors.
`pattern`	quanteda::pattern to select words to apply `weights`.
`group_data`	if `TRUE`, apply `dfm_group(x)` before creating document vectors.
`...`	additional arguments passed to quanteda::object2id.

Value

Returns a textmodel_docvector object with the following elements:

`values`	a list of matrices for word and document vectors.
`dim`	the size of the document vectors.
`concatenator`	the concatenator in `x`.
`docvars`	document variables copied from `x`.
`normalize`	if the document vectors are normalized.
`call`	the command used to execute the function.
`version`	the version of the wordvector package.

`data_corpus_news2014` *Yahoo News summaries from 2014*

Description

A corpus object containing 2,000 news summaries collected from Yahoo News via RSS feeds in 2014. The title and description of the summaries are concatenated.

Usage

```
data_corpus_news2014
```

Format

An object of class `corpus` (inherits from `character`) of length 20000.

Source

https://www.yahoo.com/news/

References

Watanabe, K. (2018). Newsmap: A semi-supervised approach to geographical news classification. Digital Journalism, 6(3), 294–309. https://doi.org/10.1080/21670811.2017.1293487

`probability` *Compute probability of words*

Description

Compute the probability of words given other words.

Usage

```
probability(
  x,
  targets,
  layer = c("words", "documents"),
  mode = c("character", "numeric"),
  ...
)
```

Arguments

`x`	a trained `textmodel_wordvector` object.
`targets`	words for which probabilities are computed.
`layer`	the layer based on which probabilities are computed.
`mode`	specify the type of resulting object.
`...`	passed to `as.matrix()`.

Value

a matrix of words or documents sorted in descending order by the probability scores when `mode = "character"`; a matrix of the probability scores when `mode = "numeric"`. When `targets` is a named numeric vector, probability scores are weighted by the values.

See Also

`similarity()`

`similarity` *Compute similarity between word or document vectors*

Description

Compute the cosine similarity between word vectors for selected words.

Usage

```
similarity(
  x,
  targets,
  layer = c("words", "documents"),
  mode = c("character", "numeric")
)
```

Arguments

`x`	a `textmodel_wordvector` object.
`targets`	words or documents for which similarity is computed.
`layer`	the layer based on which similarity is computed. This must be "documents" when `targets` are document names.
`mode`	specify the type of resulting object.

Value

a `matrix` of cosine similarity scores when `mode = "numeric"` or of words sorted in descending order by the similarity scores when `mode = "character"`. When `targets` is a named numeric vector, word (or document) vectors are weighted and summed before computing similarity scores.

See Also

`probability()`

textmodel_doc2vec *Doc2vec model*

Description

Train a doc2vec model (Le & Mikolov, 2014) using a quanteda::tokens object.

Usage

```
textmodel_doc2vec(
  x,
  dim = 50,
  type = c("dm", "dbow"),
  min_count = 5,
  window = 5,
  iter = 10,
  alpha = 0.05,
  model = NULL,
  use_ns = TRUE,
  ns_size = 5,
  sample = 0.001,
  tolower = TRUE,
  include_data = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	a quanteda::tokens or quanteda::tokens_xptr object.
`dim`	the size of the word vectors.
`type`	the architecture of the model; either "dm" (distributed memory) or "dbow" (distributed bag-of-words).
`min_count`	the minimum frequency of the words. Words less frequent than this in `x` are removed before training.
`window`	the size of the window for context words. Ignored when `type = "dbow"` as its context window is the entire document (sentence or paragraph).
`iter`	the number of iterations in model training.
`alpha`	the initial learning rate.
`model`	a trained Word2vec model; if provided, its word vectors are updated for `x`.
`use_ns`	if `TRUE`, negative sampling is used. Otherwise, hierarchical softmax is used.
`ns_size`	the size of negative samples. Only used when `use_ns = TRUE`.
`sample`	the rate of sampling of words based on their frequency. Sampling is disabled when `sample = 1.0`

`tolower`	lower-case all the tokens before fitting the model.
`include_data`	if `TRUE`, the resulting object includes the data supplied as `x`.
`verbose`	if `TRUE`, print the progress of training.
`...`	additional arguments.

Value

Returns a textmodel_doc2vec object with matrices for word and document vector values in `values`. Other elements are the same as textmodel_word2vec.

References

Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents (No. arXiv:1405.4053). arXiv. https://doi.org/10.48550/arXiv.1405.4053

`textmodel_lsa` *Latent Semantic Analysis model*

Description

Train a Latent Semantic Analysis model (Deerwester et al., 1990) on a quanteda::tokens object.

Usage

```
textmodel_lsa(
  x,
  dim = 50,
  min_count = 5L,
  engine = c("RSpectra", "irlba", "rsvd"),
  weight = "count",
  tolower = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	a quanteda::tokens or quanteda::tokens_xptr object.
`dim`	the size of the word vectors.
`min_count`	the minimum frequency of the words. Words less frequent than this in `x` are removed before training.
`engine`	select the engine perform SVD to generate word vectors.
`weight`	weighting scheme passed to `quanteda::dfm_weight()`.
`tolower`	if `TRUE` lower-case all the tokens before fitting the model.
`verbose`	if `TRUE`, print the progress of training.
`...`	additional arguments.

Value

Returns a textmodel_wordvector object with the following elements:

`values`	a matrix for word vectors values.
`weights`	a matrix for word vectors weights.
`frequency`	the frequency of words in `x`.
`engine`	the SVD engine used.
`weight`	weighting scheme.
`min_count`	the value of min_count.
`concatenator`	the concatenator in `x`.
`call`	the command used to execute the function.
`version`	the version of the wordvector package.

References

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. JASIS, 41(6), 391–407.

Examples

```
library(quanteda)
library(wordvector)

# pre-processing
corp <- corpus_reshape(data_corpus_news2014)
toks <- tokens(corp, remove_punct = TRUE, remove_symbols = TRUE) %>%
   tokens_remove(stopwords("en", "marimo"), padding = TRUE) %>%
   tokens_select("^[a-zA-Z-]+$", valuetype = "regex", case_insensitive = FALSE,
                 padding = TRUE) %>%
   tokens_tolower()

# train LSA
lsa <- textmodel_lsa(toks, dim = 50, min_count = 5, verbose = TRUE)

# find similar words
head(similarity(lsa, c("berlin", "germany", "france"), mode = "words"))
head(similarity(lsa, c("berlin" = 1, "germany" = -1, "france" = 1), mode = "values"))
head(similarity(lsa, analogy(~ berlin - germany + france)))
```

`textmodel_word2vec` *Word2vec model*

Description

Train a word2vec model (Mikolov et al., 2013) using a quanteda::tokens object.

Usage

```
textmodel_word2vec(
  x,
  dim = 50,
  type = c("cbow", "sg", "dm"),
  min_count = 5,
  window = ifelse(type == "sg", 10, 5),
  iter = 10,
  alpha = 0.05,
  model = NULL,
  use_ns = TRUE,
  ns_size = 5,
  sample = 0.001,
  tolower = TRUE,
  include_data = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	a quanteda::tokens or quanteda::tokens_xptr object.
`dim`	the size of the word vectors.
`type`	the architecture of the model; either "cbow" (continuous back-of-words), "sg" (skip-gram), or "dm" (distributed memory).
`min_count`	the minimum frequency of the words. Words less frequent than this in `x` are removed before training.
`window`	the size of the word window. Words within this window are considered to be the context of a target word.
`iter`	the number of iterations in model training.
`alpha`	the initial learning rate.
`model`	a trained Word2vec model; if provided, its word vectors are updated for `x`.
`use_ns`	if `TRUE`, negative sampling is used. Otherwise, hierarchical softmax is used.
`ns_size`	the size of negative samples. Only used when `use_ns = TRUE`.
`sample`	the rate of sampling of words based on their frequency. Sampling is disabled when `sample = 1.0`

tolower	lower-case all the tokens before fitting the model.
include_data	if TRUE, the resulting object includes the data supplied as x.
verbose	if TRUE, print the progress of training.
...	additional arguments.

Details

If type = "dm", it trains a doc2vec model but saves only word vectors to save storage space. textmodel_doc2vec should be used to access document vectors.

Users can changed the number of processors used for the parallel computing via `options(wordvector_threads)`.

Value

Returns a textmodel_word2vec object with the following elements:

values	a list of a matrix for word vector values.
weights	a matrix for word vector weights.
dim	the size of the word vectors.
type	the architecture of the model.
frequency	the frequency of words in x.
window	the size of the word window.
iter	the number of iterations in model training.
alpha	the initial learning rate.
use_ns	the use of negative sampling.
ns_size	the size of negative samples.
min_count	the value of min_count.
concatenator	the concatenator in x.
data	the original data supplied as x if include_data = TRUE.
call	the command used to execute the function.
version	the version of the wordvector package.

References

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. https://arxiv.org/abs/1310.4546.

Examples

```
library(quanteda)
library(wordvector)

# pre-processing
corp <- data_corpus_news2014
toks <- tokens(corp, remove_punct = TRUE, remove_symbols = TRUE) %>%
   tokens_remove(stopwords("en", "marimo"), padding = TRUE) %>%
   tokens_select("^[a-zA-Z-]+$", valuetype = "regex", case_insensitive = FALSE,
                 padding = TRUE) %>%
   tokens_tolower()

# train word2vec
wov <- textmodel_word2vec(toks, dim = 50, type = "cbow", min_count = 5, sample = 0.001)

# find similar words
head(similarity(wov, c("berlin", "germany", "france"), mode = "words"))
head(similarity(wov, c("berlin" = 1, "germany" = -1, "france" = 1), mode = "values"))
head(similarity(wov, analogy(~ berlin - germany + france), mode = "words"))
```

Index