

# Package ‘wordvector’

September 10, 2026

**Type** Package

**Title** Word and Document Vector Models

**Version** 0.6.4

**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/arXiv.1406.2661>, and Latent Semantic Analysis (Deerwester et al., 1990) <doi:10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-AS11%3E3.0.CO;2-9>.

**URL** <https://github.com/koheiw/wordvector>,  
<https://koheiw.github.io/wordvector/>

**License** Apache License (>= 2.0)

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

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

**Suggests** testthat, word2vec, spelling, rmarkdown, knitr

**LinkingTo** Rcpp, quanteda

**Language** en-US

**LazyData** true

**Config/roxygen2/version** 8.0.0

**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** 2026-09-10 14:40:02 UTC

Contents

|                                            |           |
|--------------------------------------------|-----------|
| analogy . . . . .                          | 2         |
| as.matrix.textmodel_doc2vec . . . . .      | 3         |
| as.textmodel_doc2vec . . . . .             | 3         |
| as.textmodel_word2vec . . . . .            | 4         |
| data_corpus_news2014 . . . . .             | 5         |
| probability . . . . .                      | 5         |
| probability.textmodel_wordvector . . . . . | 6         |
| similarity . . . . .                       | 7         |
| similarity.textmodel_wordvector . . . . .  | 7         |
| textmodel_doc2vec . . . . .                | 8         |
| textmodel_lsa . . . . .                    | 9         |
| textmodel_word2vec . . . . .               | 11        |
| <b>Index</b>                               | <b>14</b> |

---

|         |                                                  |
|---------|--------------------------------------------------|
| analogy | <i>Convert formula to named character vector</i> |
|---------|--------------------------------------------------|

---

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", padding = FALSE, ...)
```

## 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.                                                                                                           |
| padding   | if TRUE, add a row with zeros before the word vectors.                                                              |

## Value

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

---

```
as.textmodel_doc2vec
```

*Create a doc2vec model*

---

## Description

Create a doc2vec model as weighted word vectors.

**Usage**

```
as.textmodel_doc2vec(x, model, normalize = FALSE, group_data = FALSE, ...)
```

**Arguments**

|            |                                                                             |
|------------|-----------------------------------------------------------------------------|
| x          | a <a href="#">quanteda::tokens</a> or <a href="#">quanteda::dfm</a> object. |
| model      | a textmodel_wordvector object.                                              |
| normalize  | if TRUE, normalized word vectors before creating document vectors.          |
| group_data | if TRUE, apply dfm_group(x) before creating document vectors.               |
| ...        | additional arguments passed to <a href="#">quanteda::object2id</a> .        |

**Value**

Returns a textmodel\_doc2vec 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.            |

---

as.textmodel\_word2vec *Create a word2vec model*

---

**Description**

Create a word2vec model from a matrix.

**Usage**

```
as.textmodel_word2vec(x, ...)
```

**Arguments**

|     |                                           |
|-----|-------------------------------------------|
| x   | a matrix containing word vectors in rows. |
| ... | additional arguments.                     |

**Value**

Returns a dummy textmodel\_word2vec object

---

|                      |                                       |
|----------------------|---------------------------------------|
| data_corpus_news2014 | <i>Yahoo News summaries from 2014</i> |
|----------------------|---------------------------------------|

---

**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 | <i>Generic function to extract probabilities</i> |
|-------------|--------------------------------------------------|

---

**Description**

Generic function to extract probabilities

**Usage**

```
probability(x, ...)
```

**Arguments**

|     |                                                   |
|-----|---------------------------------------------------|
| x   | an object from which probabilities are extracted. |
| ... | passed to underlying functions.                   |

---

```
probability.textmodel_wordvector
```

*Compute probability of words*

---

## Description

Compute the probability of words given other words.

## Usage

```
## S3 method for class 'textmodel_wordvector'
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 <code>as.matrix()</code> .                 |

## 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 | <i>Generic function to extract similarities</i> |
|------------|-------------------------------------------------|

---

**Description**

Generic function to extract similarities

**Usage**

```
similarity(x, ...)
```

**Arguments**

|     |                                                  |
|-----|--------------------------------------------------|
| x   | an object from which similarities are extracted. |
| ... | passed to underlying functions.                  |

---

|                                 |                                                            |
|---------------------------------|------------------------------------------------------------|
| similarity.textmodel_wordvector | <i>Compute similarity between word or document vectors</i> |
|---------------------------------|------------------------------------------------------------|

---

**Description**

Compute the cosine similarity between word vectors for selected words.

**Usage**

```
## S3 method for class 'textmodel_wordvector'
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.                                                                      |
| ...     | not used.                                                                                                  |

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 | <i>Doc2vec model</i> |
|-------------------|----------------------|

---

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 <a href="#">quanteda::tokens</a> or <a href="#">quanteda::tokens_xptr</a> 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). |

|                           |                                                                                                             |
|---------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>iter</code>         | the number of iterations in model training.                                                                 |
| <code>alpha</code>        | the initial learning rate.                                                                                  |
| <code>model</code>        | a trained Word2vec model; if provided, its word vectors are updated for <code>x</code> .                    |
| <code>use_ns</code>       | if TRUE, negative sampling is used. Otherwise, hierarchical softmax is used.                                |
| <code>ns_size</code>      | the size of negative samples. Only used when <code>use_ns = TRUE</code> .                                   |
| <code>sample</code>       | the rate of sampling of words based on their frequency. Sampling is disabled when <code>sample = 1.0</code> |
| <code>tolower</code>      | lower-case all the tokens before fitting the model.                                                         |
| <code>include_data</code> | if TRUE, the resulting object includes the data supplied as <code>x</code> .                                |
| <code>verbose</code>      | if TRUE, print the progress of training.                                                                    |
| <code>...</code>          | additional arguments.                                                                                       |

### Value

Returns a `textmodel_doc2vec` object with matrices for word and document vector values, `quanteda::docvars` and `quanteda::ntoken` of `x`. 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>

---

|                            |                                       |
|----------------------------|---------------------------------------|
| <code>textmodel_lsa</code> | <i>Latent Semantic Analysis model</i> |
|----------------------------|---------------------------------------|

---

### 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**

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

**Value**

Returns a `textmodel_wordvector` object with the following elements:

|                           |                                            |
|---------------------------|--------------------------------------------|
| <code>values</code>       | a matrix for word vectors values.          |
| <code>weights</code>      | a matrix for word vectors weights.         |
| <code>frequency</code>    | the frequency of words in <code>x</code> . |
| <code>engine</code>       | the SVD engine used.                       |
| <code>weight</code>       | weighting scheme.                          |
| <code>min_count</code>    | the value of <code>min_count</code> .      |
| <code>concatenator</code> | the concatenator in <code>x</code> .       |
| <code>call</code>         | the command used to execute the function.  |
| <code>version</code>      | 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 | <i>Word2vec model</i> |
|--------------------|-----------------------|

---

## 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

|                        |                                                                                                                          |
|------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>         | a <a href="#">quanteda::tokens</a> or <a href="#">quanteda::tokens_xptr</a> object.                                      |
| <code>dim</code>       | the size of the word vectors.                                                                                            |
| <code>type</code>      | the architecture of the model; either "cbow" (continuous back-of-words), "sg" (skip-gram), or "dm" (distributed memory). |
| <code>min_count</code> | the minimum frequency of the words. Words less frequent than this in <code>x</code> are removed before training.         |
| <code>window</code>    | the size of the word window. Words within this window are considered to be the context of a target word.                 |
| <code>iter</code>      | the number of iterations in model training.                                                                              |
| <code>alpha</code>     | 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)`. When the value is large than one, the result of every execution becomes slightly different even if `set.seed()` is used because parameters are updated in different orders by the processors.

### 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(quantda)
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

## \* datasets

data\_corpus\_news2014, [5](#)

analogy, [2](#)

as.matrix.textmodel\_doc2vec, [3](#)

as.matrix.textmodel\_word2vec  
(as.matrix.textmodel\_doc2vec),  
[3](#)

as.textmodel\_doc2vec, [3](#)

as.textmodel\_word2vec, [4](#)

data\_corpus\_news2014, [5](#)

formula, [2](#)

probability, [5](#)

probability(), [8](#)

probability.textmodel\_wordvector, [6](#)

quanteda::dfm, [4](#)

quanteda::dfm\_weight(), [10](#)

quanteda::docvars, [9](#)

quanteda::ntoken, [9](#)

quanteda::object2id, [4](#)

quanteda::tokens, [4](#), [8–11](#)

quanteda::tokens\_xptr, [8](#), [10](#), [11](#)

similarity, [7](#)

similarity(), [2](#), [6](#)

similarity.textmodel\_wordvector, [7](#)

textmodel\_doc2vec, [8](#), [12](#)

textmodel\_lsa, [9](#)

textmodel\_word2vec, [9](#), [11](#)