Package ‘wordvector’

January 7, 2025

Type Package

Title Word and Document Vector Models

Version 0.2.0

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

Description Create dense vector representation of words and documents using 'quanteda'. Currently implements Word2vec (Mikolov et al., 2013) <doi:10.48550/arXiv.1310.4546> and Latent 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 7.3.2

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]
(<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 2025-01-07 22:30:02 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_wordvector` *Extract word vectors*

Description

Extract word vectors from a `textmodel_wordvector` or `textmodel_docvector` object.

Usage

```
## S3 method for class 'textmodel_wordvector'
as.matrix(x, ...)
```

Arguments

`x`	a `textmodel_wordvector` or `textmodel_docvector` object.
`...`	not used

Value

a matrix that contain the word vectors in rows

`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

similarity *Compute similarity between word vectors*

Description

Compute cosine similarity between word vectors for selected words.

Usage

```
similarity(x, words, mode = c("words", "values"))
```

Arguments

x	a textmodel_wordvector object.
words	words for which similarity is computed.
mode	specify the type of resulting object.

Value

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

See Also

analogy()

textmodel_doc2vec *Create distributed representation of documents*

Description

Create distributed representation of documents as weighted word vectors.

Usage

```
textmodel_doc2vec(x, model = NULL, ...)
```

Arguments

x	a quanteda::tokens object.
model	a textmodel_wordvector object.
...	passed to [word2vec] when model = NULL.

Value

Returns a textmodel_docvector object with elements inherited from `model` or passed via `...` plus:

`values` a matrix for document vectors.

`call` the command used to execute the function.

`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",
  verbose = FALSE,
  ...
)
```

Arguments

`x` a quanteda::tokens 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()`.

`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.
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., 2023) in different architectures on a quanteda::tokens object.

Usage

```
textmodel_word2vec(
  x,
  dim = 50,
  type = c("cbow", "skip-gram"),
  min_count = 5L,
  window = ifelse(type == "cbow", 5L, 10L),
  iter = 10L,
  alpha = 0.05,
  use_ns = TRUE,
  ns_size = 5L,
  sample = 0.001,
  normalize = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`x` a quanteda::tokens object.

`dim` the size of the word vectors.

`type` the architecture of the model; either "cbow" (continuous back of words) or "skip-gram".

`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.

`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`

`normalize` if `TRUE`, normalize the vectors in `values` and `weights`.

`verbose` if `TRUE`, print the progress of training.

`...` additional arguments.

Details

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

Value

Returns a textmodel_wordvector object with the following elements:

`values` 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.
`concatenator`	the concatenator in `x`.
`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
w2v <- textmodel_word2vec(toks, dim = 50, type = "cbow", min_count = 5, sample = 0.001)

# find similar words
head(similarity(w2v, c("berlin", "germany", "france"), mode = "words"))
head(similarity(w2v, c("berlin" = 1, "germany" = -1, "france" = 1), mode = "values"))
head(similarity(w2v, analogy(~ berlin - germany + france), mode = "words"))
```

`weights` *[experimental] Extract word vector weights*

Description

[experimental] Extract word vector weights

Usage

```
weights(x, mode = c("words", "values"))
```

Arguments

`x`	a `textmodel_wordvector` object.
`mode`	specify the type of resulting object.

Value

a `matrix` of word vector weights when `mode = "value"` or of words sorted in descending order by the weights when `mode = "word"`.

Index