Package: wordvector (via r-universe)

December 19, 2024

Type Package

Title Word and Document Vector Models

Version 0.1.1

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

Roxygen list(markdown = TRUE)

Language en-US

LazyData true

Config/pak/sysreqs libicu-dev libxml2-dev

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

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

RemoteRef HEAD

RemoteSha c8914906ccfbd488c34e98a145d856a5fc257b38

Contents

`analogy` *[experimental] Find analogical relationships between words*

Description

[experimental] Find analogical relationships between words

Usage

```
analogy(x, formula, n = 10, exclude = TRUE, type = c("word", "simil"))
```

Arguments

`x`	a `textmodel_wordvector` object.
`formula`	a formula object that defines the relationship between words using + or - operators.
`n`	the number of words in the resulting object.
`exclude`	if `TRUE`, words in `formula` are excluded from the result.
`type`	specify the type of vectors to be used. "word" is word vectors while "simil" is similarity vectors.

Value

a `data.frame` with the words sorted and their cosine similarity sorted in descending order.

References

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

Examples

```
## Not run:
# from Mikolov et al. (2023)
analogy(wdv, ~ berlin - germany + france)
analogy(wdv, ~ quick - quickly + slowly)

## End(Not run)
```

---

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

---

doc2vec *Create distributed representation of documents*

Description

Create distributed representation of documents

Usage

```
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:

vectors	a matrix for document vectors.
call	the command used to execute the function.

---

lsa *Latent Semantic Analysis model*

Description

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

Usage

```
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:

vectors	a matrix for word vectors.
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 <- lsa(toks, dim = 50, min_count = 5, verbose = TRUE, )
head(similarity(lsa, c("berlin", "germany", "france"), mode = "word"))
analogy(lsa, ~ berlin - germany + france)
```

`similarity` *Compute similarity between word vectors*

Description

Compute similarity between word vectors

Usage

```
similarity(x, words, mode = c("simil", "word"))
```

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 = "simil"` or of words sorted by the similarity scores when `mode = "word`.

`word2vec` *Word2vec model*

Description

Train a Word2vec model (Mikolov et al., 2023) in different architectures on a quanteda::tokens object.

Usage

```
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,
  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
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:

vectors	a matrix for word vectors.
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 <- word2vec(toks, dim = 50, type = "cbow", min_count = 5, sample = 0.001)
head(similarity(w2v, c("berlin", "germany", "france"), mode = "word"))
analogy(w2v, ~ berlin - germany + france)
```

Index