Package 'LSX'

September 18, 2021

Type Package

Title Model for Semisupervised Text Analysis Based on Word Embeddings

Date 2021-09-18

Version 1.0.2

Description A word embeddings-based semisupervised model for document scaling Watanabe (2020) <doi:10.1080/19312458.2020.1832976>.
LSS allows users to analyze large and complex corpora on arbitrary dimensions with seed words exploiting efficiency of word embeddings (SVD, Glove).
It can generate word vectors on a users-provided corpus or incorporate a pre-trained word vectors.

License GPL-3

LazyData TRUE

Encoding UTF-8

Depends methods, R (>= 3.5.0)

Imports quanteda (>= 2.0), quanteda.textstats, stringi, digest, Matrix, RSpectra, irlba, rsvd, rsparse, proxyC, stats, ggplot2, ggrepel, reshape2, locfit

Suggests testthat

RoxygenNote 7.1.1

BugReports https://github.com/koheiw/LSX/issues

NeedsCompilation no

Author Kohei Watanabe [aut, cre, cph]

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

Repository CRAN

Date/Publication 2021-09-18 08:00:02 UTC

R topics documented:

`as.seedwords` *Convenient function to convert a list to seed words*

Description

Convenient function to convert a list to seed words

Usage

```
as.seedwords(x, upper = 1, lower = 2, concatenator = "_")
```

Arguments

`x`	a list of characters vectors or a dictionary object
`upper`	numeric index or key for seed words for higher scores
`lower`	numeric index or key for seed words for lower scores
`concatenator`	character to replace separators of multi-word seed words

Value

named numeric vector for seed words with polarity scores

cohesion *Computes cohesion of components of latent semantic analysis*

Description

Computes cohesion of components of latent semantic analysis

Usage

```
cohesion(object, bandwidth = 10)
```

Arguments

object	a fitted textmodel_lss
bandwidth	size of window for smoothing

data_dictionary_ideology *Seed words for analysis of left-right political ideology*

Description

Seed words for analysis of left-right political ideology

Examples

```
as.seedwords(data_dictionary_ideology)
```

data_dictionary_sentiment *Seed words for analysis of positive-negative sentiment*

Description

Seed words for analysis of positive-negative sentiment

References

Turney, P. D., & Littman, M. L. (2003). Measuring Praise and Criticism: Inference of Semantic Orientation from Association. ACM Trans. Inf. Syst., 21(4), 315–346. doi: 10.1145/944012.944013

Examples

```
as.seedwords(data_dictionary_sentiment)
```

`data_textmodel_lss_russianprotests` *A fitted LSS model on street protest in Russia*

Description

This model was trained on a Russian media corpus (newspapers, TV transcripts and newswires) to analyze framing of street protests. The scale is protests as "freedom of expression" (high) vs "social disorder" (low). Although some slots are missing in this object (because the model was imported from the original Python implementation), it allows you to scale texts using `predict`.

References

Lankina, Tomila, and Kohei Watanabe. "'Russian Spring' or 'Spring Betrayal'? The Media as a Mirror of Putin's Evolving Strategy in Ukraine." Europe-Asia Studies 69, no. 10 (2017): 1526–56. doi: 10.1080/09668136.2017.1397603.

`diagnosys` *Identify noisy documents in a corpus*

Description

Identify noisy documents in a corpus

Usage

```
diagnosys(x, ...)
```

Arguments

`x`	character or corpus object whose texts will be diagnosed
`...`	extra arguments passed to `tokens`

predict.textmodel_lss *Prediction method for textmodel_lss*

Description

Prediction method for textmodel_lss

Usage

```
## S3 method for class 'textmodel_lss'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  density = FALSE,
  rescaling = TRUE,
  ...
)
```

Arguments

object	a fitted LSS textmodel
newdata	dfm on which prediction should be made
se.fit	if TRUE, it returns standard error of document scores.
density	if TRUE, returns frequency of model terms in documents. Density distribution of model terms can be used to remove documents about unrelated subjects.
rescaling	if TRUE, scores are normalized using scale().
...	not used

seedwords *Seed words for Latent Semantic Analysis*

Description

Seed words for Latent Semantic Analysis

Usage

```
seedwords(type)
```

Arguments

type	type of seed words currently only for sentiment (sentiment) or political ideology (ideology).

References

Turney, P. D., & Littman, M. L. (2003). Measuring Praise and Criticism: Inference of Semantic Orientation from Association. ACM Trans. Inf. Syst., 21(4), 315–346. doi: 10.1145/944012.944013

Examples

```
seedwords('sentiment')
```

`smooth_lss` *Smooth predicted LSS scores by local polynomial regression*

Description

Smooth predicted LSS scores by local polynomial regression

Usage

```
smooth_lss(
  x,
  lss_var = "fit",
  date_var = "date",
  span = 0.1,
  from = NULL,
  to = NULL,
  engine = c("loess", "locfit"),
  ...
)
```

Arguments

`x`	a `data.frame` containing LSS scores and dates
`lss_var`	the name of the column for LSS scores
`date_var`	the name of the columns for dates
`span`	determines the level of smoothing.
`from`	start of the time period
`to`	end of the time period
`engine`	specifies the function to smooth LSS scores: `loess()` or `locfit()`. The latter should be used when n > 10000.
`...`	extra arguments passed to `loess()` or `lp()`

`textmodel_lss` *A word embeddings-based semisupervised model for document scaling*

Description

A word embeddings-based semisupervised model for document scaling

Usage

```
textmodel_lss(x, ...)

## S3 method for class 'dfm'
textmodel_lss(
  x,
  seeds,
  terms = NULL,
  k = 300,
  slice = NULL,
  weight = "count",
  cache = FALSE,
  simil_method = "cosine",
  engine = c("RSpectra", "irlba", "rsvd"),
  auto_weight = FALSE,
  include_data = FALSE,
  verbose = FALSE,
  ...
)

## S3 method for class 'fcm'
textmodel_lss(
  x,
  seeds,
  terms = NULL,
  w = 50,
  max_count = 10,
  weight = "count",
  cache = FALSE,
  simil_method = "cosine",
  engine = c("rsparse"),
  auto_weight = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

x	a dfm or fcm created by `quanteda::dfm()` or `quanteda::fcm()`

...	additional arguments passed to the underlying engine.
seeds	a character vector, named numeric vector that contains seed words. If seed words contain "*", they are interpreted as glob patterns. See quanteda::valuetype.
terms	words weighted as model terms. All the features of quanteda::dfm() or quanteda::fcm() will be used if not specified.
k	the number of singular values requested to the SVD engine. Only used when x is a dfm.
slice	a number or indices of the components of word vectors used to compute similarity; slice < k to truncate word vectors; useful for diagnosys and simulation.
weight	weighting scheme passed to quanteda::dfm_weight(). Ignored when engine is "rsparse".
cache	if TRUE, save result of SVD for next execution with identical x and settings. Use the base::options(lss_cache_dir) to change the location cache files to be save.
simil_method	specifies method to compute similarity between features. The value is passed to quanteda.textstats::textstat_simil(), "cosine" is used otherwise.
engine	select the engine to factorize x to get word vectors. Choose from RSpectra::svds(), irlba::irlba(), rsvd::rsvd(), and rsparse::GloVe().
auto_weight	automatically determine weights to approximate the polarity of terms to seed words. See details.
include_data	if TRUE, fitted model include the dfm supplied as x.
verbose	show messages if TRUE.
w	the size of word vectors. Used only when x is a fcm.
max_count	passed to x_max in rsparse::GloVe$new() where cooccurrence counts are ceiled to this threshold. It should be changed according to the size of the corpus. Used only when x is a fcm.

Details

Latent Semantic Scaling (LSS) is a semisupervised document scaling method. textmodel_lss() constructs word vectors from use-provided documents (x) and weights words (terms) based on their semantic proximity to seed words (seeds). Seed words are any known polarity words (e.g. sentiment words) that users should manually choose. The required number of seed words are usually 5 to 10 for each end of the scale.

If seeds is a named numeric vector with positive and negative values, a bipolar LSS model is construct; if seeds is a character vector, a unipolar LSS model. Usually bipolar models perform better in document scaling because both ends of the scale are defined by the user.

A seed word's polarity score computed by textmodel_lss() tends to diverge from its original score given by the user because it's score is affected not only by its original score but also by the original scores of all other seed words. If auto_weight = TRUE, the original scores are weighted automatically using stats::optim() to minimize the squared difference between seed words' computed and original scores. Weighted scores are saved in seed_weighted in the object.

References

Watanabe, Kohei. 2020. "Latent Semantic Scaling: A Semisupervised Text Analysis Technique for New Domains and Languages", Communication Methods and Measures. doi: 10.1080/19312458.2020.1832976.

Watanabe, Kohei. 2017. "Measuring News Bias: Russia's Official News Agency ITAR-TASS' Coverage of the Ukraine Crisis" European Journal of Communication. doi: 10.1177/0267323117695735.

Examples

```
library("quanteda")
con <- url("https://bit.ly/2GZwLcN", "rb")
corp <- readRDS(con)
close(con)
toks <- corpus_reshape(corp, "sentences") %>%
        tokens(remove_punct = TRUE) %>%
        tokens_remove(stopwords("en")) %>%
        tokens_select("^[\\p{L}]+$", valuetype = "regex", padding = TRUE)
dfmt <- dfm(toks) %>%
        dfm_trim(min_termfreq = 10)

seed <- as.seedwords(data_dictionary_sentiment)

# SVD
lss_svd <- textmodel_lss(dfmt, seed)
summary(lss_svd)

# sentiment model on economy
eco <- head(char_keyness(toks, 'econom*'), 500)
svd_eco <- textmodel_lss(dfmt, seed, terms = eco)

# sentiment model on politics
pol <- head(char_keyness(toks, 'politi*'), 500)
svd_pol <- textmodel_lss(dfmt, seed, terms = pol)

# GloVe
fcmt  <- fcm(toks, context = "window", count = "weighted", weights = 1 / (1:5), tri = TRUE)
lss_glov <- textmodel_lss(fcmt, seed)
summary(lss_glov)
```

---

textplot_simil *Plot similarity between seed words*

Description

Plot similarity between seed words

Usage

```
textplot_simil(x)
```

Arguments

x	fitted textmodel_lss object

---

textplot_terms *Plot polarity scores of words*

Description

Plot polarity scores of words

Usage

```
textplot_terms(x, highlighted = NULL, max_words = 10000)
```

Arguments

x	a fitted textmodel_lss object.
highlighted	quanteda::pattern to select words to highlight.
max_words	the maximum number of words to plot. Words are randomly sampled to keep the number below the limit.

---

textstat_context *Identify context words using user-provided patterns*

Description

Identify context words using user-provided patterns

Usage

```
textstat_context(
  x,
  pattern,
  valuetype = c("glob", "regex", "fixed"),
  case_insensitive = TRUE,
  window = 10,
  min_count = 10,
  remove_pattern = TRUE,
  n = 1,
  skip = 0,
  ...
```

```
)

char_context(
  x,
  pattern,
  valuetype = c("glob", "regex", "fixed"),
  case_insensitive = TRUE,
  window = 10,
  min_count = 10,
  remove_pattern = TRUE,
  p = 0.001,
  n = 1,
  skip = 0
)

char_keyness(
  x,
  pattern,
  valuetype = c("glob", "regex", "fixed"),
  case_insensitive = TRUE,
  window = 10,
  min_count = 10,
  remove_pattern = TRUE,
  p = 0.001,
  n = 1,
  skip = 0
)
```

Arguments

x	a tokens object created by quanteda::tokens().
pattern	quanteda::pattern() to specify target words.
valuetype	the type of pattern matching: "glob" for "glob"-style wildcard expressions; "regex" for regular expressions; or "fixed" for exact matching. See quanteda::valuetype() for details.
case_insensitive	if TRUE, ignore case when matching.
window	size of window for collocation analysis.
min_count	minimum frequency of words within the window to be considered as collocations.
remove_pattern	if TRUE, keywords do not contain target words.
n	integer vector specifying the number of elements to be concatenated in each ngram. Each element of this vector will define a n in the n-gram(s) that are produced.
skip	integer vector specifying the adjacency skip size for tokens forming the ngrams, default is 0 for only immediately neighbouring words. For skipgrams, skip can be a vector of integers, as the "classic" approach to forming skip-grams is to set

skip = k where k is the distance for which k or fewer skips are used to construct the n-gram. Thus a "4-skip-n-gram" defined as `skip = 0:4` produces results that include 4 skips, 3 skips, 2 skips, 1 skip, and 0 skips (where 0 skips are typical n-grams formed from adjacent words). See Guthrie et al (2006).

`...` additional arguments passed to `textstat_keyness()`.

p threshold for statistical significance of collocations.

See Also

`tokens_select()` and `textstat_keyness()`

Examples

```
#' @examples

require(quanteda)
con <- url("https://bit.ly/2GZwLcN", "rb")
corp <- readRDS(con)
close(con)
corp <- corpus_reshape(corp, 'sentences')
toks <- tokens(corp, remove_punct = TRUE)
toks <- tokens_remove(toks, stopwords("en"))

# economy keywords
eco <- char_context(toks, 'econom*')
head(eco, 20)

tstat_eco <- textstat_context(toks, 'econom*')
head(tstat_eco)

# politics keywords
pol <- char_context(toks, 'politi*')
head(pol, 20)

# politics keywords
tstat_pol <- textstat_context(toks, 'politi*')
head(tstat_pol)
```

Index