Package 'LSX'

September 12, 2025

Type Package

Title Semi-Supervised Algorithm for Document Scaling

Version 1.5.0

Description A word embeddings-based semi-supervised 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 R (>= 3.5.0)

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

Suggests testthat, spelling, knitr, rmarkdown, wordvector, irlba, rsvd, rsparse

RoxygenNote 7.3.2

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

URL https://koheiw.github.io/LSX/

Language en-US

NeedsCompilation no

Author Kohei Watanabe [aut, cre, cph]

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

Repository CRAN

Date/Publication 2025-09-12 08:20:02 UTC

Contents

`as.seedwords` *Convert a list or a dictionary to seed words*

Description

Convert a list or a dictionary 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

`as.textmodel_lss` *Create a Latent Semantic Scaling model from various objects*

Description

Create a new textmodel_lss object from an existing or foreign objects.

Usage

```
as.textmodel_lss(x, ...)

## S3 method for class 'matrix'
as.textmodel_lss(
  x,
  seeds,
  terms = NULL,
  slice = NULL,
  simil_method = "cosine",
  auto_weight = FALSE,
  verbose = FALSE,
  ...
)

## S3 method for class 'numeric'
as.textmodel_lss(x, ...)

## S3 method for class 'textmodel_lss'
as.textmodel_lss(x, ...)

## S3 method for class 'textmodel_wordvector'
as.textmodel_lss(x, seeds, terms = NULL, verbose = FALSE, spatial = TRUE, ...)
```

Arguments

`x`	an object from which a new textmodel_lss object is created. See details.
`...`	arguments used to create a new object. `seeds` must be given when `x` is a dense matrix or a fitted textmodel_lss.
`seeds`	a character vector or named numeric vector that contains seed words. If seed words contain "*", they are interpreted as glob patterns. See quanteda::valuetype.
`terms`	a character vector or named numeric vector that specify words for which polarity scores will be computed; if a numeric vector, words' polarity scores will be weighted accordingly; if `NULL`, all the features in `x` except those less frequent than `min_count` will be used.
`slice`	a number or indices of the components of word vectors used to compute similarity; `slice` < `k` to further truncate word vectors; useful for diagnosys and simulation.

simil_method	specifies method to compute similarity between features. The value is passed to quanteda.textstats::textstat_simil(), "cosine" is used otherwise.
auto_weight	automatically determine weights to approximate the polarity of terms to seed words. Deprecated.
verbose	show messages if TRUE.
spatial	if TRUE, return a spatial model. Otherwise, a probabilistic model.

Details

If x is a textmodel_lss, original word vectors are reused to compute polarity scores with new seed words. It is also possible to subset word vectors via slice if it was trained originally using SVD.

If x is a dense matrix, it is treated as a column-oriented word vectors with which polarity of words are computed. If x is a named numeric vector, the values are treated as polarity scores of the words in the names.

Value

a dummy textmodel_lss object

---

bootstrap_lss *[experimental] Compute polarity scores with different hyper-parameters*

Description

A function to compute polarity scores of words and documents by resampling hyper-parameters from a fitted LSS model.

Usage

```
bootstrap_lss(
  x,
  what = c("seeds", "k"),
  mode = c("terms", "coef", "predict"),
  remove = FALSE,
  from = 100,
  to = NULL,
  by = 50,
  verbose = FALSE,
  ...
)
```

Arguments

x	a fitted textmodel_lss object.
what	choose the hyper-parameter to resample in bootstrapping.
mode	choose the type of the result of bootstrapping. If coef, returns the polarity scores of words; if terms, returns words sorted by the polarity scores in descending order; if predict, returns the polarity scores of documents.
remove	if TRUE, remove each seed word when what = "seeds".
from, to, by	passed to seq() to generate values for k; only used when what = "k".
verbose	show messages if TRUE.
...	additional arguments passed to as.textmodel_lss() and predict().

Details

bootstrap_lss() creates LSS fitted textmodel_lss objects internally by resampling hyper-parameters and computes polarity of words or documents. The resulting matrix can be used to asses the validity and the reliability of seeds or k.

Note that the objects created by as.textmodel_lss() does not contain data, users must pass newdata via ... when mode = "predict".

---

coef.textmodel_lss *Extract model coefficients from a fitted textmodel_lss object*

Description

coef() extract model coefficients from a fitted textmodel_lss object. coefficients() is an alias.

Usage

```
## S3 method for class 'textmodel_lss'
coef(object, ...)

coefficients.textmodel_lss(object, ...)
```

Arguments

object	a fitted textmodel_lss object.
...	not used.

`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.

`optimize_lss` *[experimental] Compute variance ratios with different hyper-parameters*

Description

[experimental] Compute variance ratios with different hyper-parameters

Usage

```
optimize_lss(x, ...)
```

Arguments

`x`	a fitted textmodel_lss object.
`...`	additional arguments passed to bootstrap_lss.

Details

`optimize_lss()` computes variance ratios with different values of hyper-parameters using bootstrap_lss. The variance ration v is defined as

$$v = \sigma^2_{documents}/\sigma^2_{words}.$$

It maximizes when the model best distinguishes between the documents on the latent scale.

Examples

```
## Not run:
# the unit of analysis is not sentences
dfmt_grp <- dfm_group(dfmt)

# choose best k
v1 <- optimize_lss(lss, what = "k", from = 50,
                   newdata = dfmt_grp, verbose = TRUE)
plot(names(v1), v1)

# find bad seed words
v2 <- optimize_lss(lss, what = "seeds", remove = TRUE,
                   newdata = dfmt_grp, verbose = TRUE)
barplot(v2, las = 2)

## End(Not run)
```

`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,
  rescale = TRUE,
  cut = NULL,
  min_n = 0L,
  ...
)
```

Arguments

`object`	a fitted LSS textmodel.
`newdata`	a dfm on which prediction should be made.
`se_fit`	if `TRUE`, returns standard error of document scores.
`density`	if `TRUE`, returns frequency of polarity words in documents.
`rescale`	if `TRUE`, normalizes polarity scores using `scale()`.
`cut`	a vector of one or two percentile values to dichotomized polarty scores of words. When two values are given, words between them receive zero polarity.
`min_n`	set the minimum number of polarity words in documents.
`...`	not used

Details

Polarity scores of documents are the means of polarity scores of words weighted by their frequency. When `se_fit = TRUE`, this function returns the weighted means, their standard errors, and the number of polarity words in the documents. When `rescale = TRUE`, it converts the raw polarity scores to z sores for easier interpretation. When `rescale = FALSE` and `cut` is used, polarity scores of documents are bounded by [-1.0, 1.0].

Documents tend to receive extreme polarity scores when they have only few polarity words. This is problematic when LSS is applied to short documents (e.g. social media posts) or individual sentences, but users can alleviate this problem by adding zero polarity words to short documents using `min_n`. This setting does not affect empty documents.

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 polarity scores*

Description

Smooth predicted polarity scores by local polynomial regression.

Usage

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

Arguments

x	a data.frame containing polarity scores and dates.
lss_var	the name of the column in x for polarity scores.
date_var	the name of the column in x for dates.
span	the level of smoothing.
groups	specify the columns in x to smooth separately by the group; the columns must be factor, character or logical.
from, to, by	the the range and the internal of the smoothed scores; passed to seq.Date.
engine	specifies the function to be used for smoothing.
...	additional arguments passed to the smoothing function.

Details

Smoothing is performed using `stats::loess()` or `locfit::locfit()`. When the x has more than 10000 rows, it is usually better to choose the latter by setting `engine = "locfit"`. In this case, span is passed to `locfit::lp(nn = span)`.

---

textmodel_lss *Fit a Latent Semantic Scaling model*

Description

Latent Semantic Scaling (LSS) is a semi-supervised algorithm for document scaling based on word embedding.

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,
  group_data = FALSE,
  verbose = FALSE,
```

```
  ...
)

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

## S3 method for class 'tokens'
textmodel_lss(
  x,
  seeds,
  terms = NULL,
  k = 200,
  min_count = 5,
  engine = "wordvector",
  tolower = TRUE,
  include_data = FALSE,
  group_data = FALSE,
  spatial = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

x	a dfm or fcm created by quanteda::dfm(), quanteda::fcm(), quanteda::tokens or quanteda::tokens_xptr object.
...	additional arguments passed to the underlying engine.
seeds	a character vector or named numeric vector that contains seed words. If seed words contain "*", they are interpreted as glob patterns. See quanteda::valuetype.
terms	a character vector or named numeric vector that specify words for which polarity scores will be computed; if a numeric vector, words' polarity scores will be weighted accordingly; if NULL, all the features in x except those less frequent than min_count will be used.
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 further truncate word vectors; useful for diagnosys and simulation.
weight	weighting scheme passed to quanteda::dfm_weight(). Ignored when engine = "rsparse".
cache	if TRUE, save the 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 generate word vectors. If x is a dfm, RSpectra::svds(), irlba::irlba() or rsvd::rsvd(). If x is a fcm, rsparse::GloVe(). If x is a tokens (or tokens_xptr), wordvector::textmodel_word2vec().
auto_weight	automatically determine weights to approximate the polarity of terms to seed words. Deprecated.
include_data	if TRUE, the fitted model includes the dfm supplied as x.
group_data	if TRUE, apply dfm_group(x) before saving the dfm.
verbose	show messages if TRUE.
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.
min_count	the minimum frequency of the words. Words less frequent than this in x are removed before training.
tolower	if TRUE, lower-case all the words in the model.
spatial	[experimental] if FALSE, return a probabilistic model. See the details.

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 model is construct; if seeds is a character vector, a unipolar 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.

When x is a tokens or tokens_xptr object, wordvector::textmodel_word2vec is called internally with type = "skip-gram" and other arguments passed via .... If spatial = TRUE, it return a spatial model; otherwise a probabilistic model. While the polarity scores of words are their cosine

similarity to seed words in spatial models, they are predicted probability that the seed words to occur in their contexts. The probabilistic models are still experimental, so use them with caution.

Please visit the package website for examples.

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.

`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_highlighted = 50,
  max_words = 1000,
  sampling = c("absolute", "relative"),
  ...
)
```

Arguments

x	a fitted textmodel_lss object.
highlighted	quanteda::pattern to select words to highlight. If a quanteda::dictionary is passed, words in the top-level categories are highlighted in different colors.
max_highlighted	the maximum number of words to highlight. When highlighted = NULL, words are randomly sampled proportionally to beta ^ 2 * log(frequency) for highlighting.
max_words	the maximum number of words to plot. Words are randomly sampled to keep the number below the limit.
sampling	if "relative", words are sampled based on their squared deviation from the mean for highlighting; if "absolute", they are sampled based on the squared distance from zero.
...	passed to underlying functions. See the Details.

Details

Users can customize the plots through . . . , which is passed to ggplot2::geom_text() and ggrepel::geom_text_repel(). The colors are specified internally but users can override the settings by appending ggplot2::scale_colour_manual() or ggplot2::scale_colour_brewer(). The legend title can also be modified using ggplot2::labs().

---

textstat_context *Identify context words*

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
)
```

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 n-gram. 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 n-grams, 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 quanteda.textstats::textstat_keyness().
p	threshold for statistical significance of collocations.

See Also

quanteda.textstats::textstat_keyness()

Index