

Package ‘LSX’

February 26, 2023

Type Package

Title Semi-Supervised Algorithm for Document Scaling

Version 1.3.1

Description A word embeddings-based semi-supervised model for document scaling Watanabe (2020) <[doi:10.1080/19312458.2020.1832976](https://doi.org/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 knitr, rmarkdown, testthat

RoxygenNote 7.2.3

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 2023-02-26 03:22:01 UTC

R topics documented:

as.seedwords	2
bootstrap_lss	2
coef.textmodel_lss	3
data_dictionary_ideology	4

data_dictionary_sentiment	4
data_textmodel_lss_russianprotests	4
predict.textmodel_lss	5
seedwords	6
smooth_lss	6
textmodel_lss	7
textplot_simil	9
textplot_terms	10
textstat_context	10

Index	12
--------------	-----------

as.seedwords	<i>Convert a list or a dictionary to seed words</i>
--------------	---

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

bootstrap_lss	<i>[experimental] Compute polarity scores with different hyper-parameters</i>
---------------	---

Description

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

Usage

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

Arguments

x	a fitted textmodel_lass object.
what	choose the hyper-parameter to resample in bootstrapping.
mode	choose the type of the result of bootstrapping. If coef, returns polarity scores; if terms, returns words sorted by the polarity scores in descending order.
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_lass().

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

Description

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

Usage

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

coefficients.textmodel_lass(object, ...)
```

Arguments

object	a fitted textmodel_lass 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.

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 <code>scale()</code> .
cut	a vector of one or two percentile values to dichotomized polarity 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 scores 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 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.
lass_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: <code>loess()</code> or <code>locfit()</code> . The latter should be used when $n > 10000$.
...	extra arguments passed to <code>loess()</code> or <code>lp()</code>

textmodel_lass	<i>Fit a Latent Semantic Scaling model</i>
----------------	--

Description

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

Usage

```
textmodel_lass(x, ...)

## S3 method for class 'dfm'
textmodel_lass(
  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_lass(
  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 <code>quanteda::dfm()</code> or <code>quanteda::fcm()</code>
...	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 <code>quanteda::valuetype</code> .
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 of <code>quanteda::dfm()</code> or <code>quanteda::fcm()</code> 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; <code>slice < k</code> to further truncate word vectors; useful for diagnosis and simulation.
weight	weighting scheme passed to <code>quanteda::dfm_weight()</code> . Ignored when engine is "rsparse".
cache	if TRUE, save result of SVD for next execution with identical x and settings. Use the <code>base::options(lss_cache_dir)</code> to change the location cache files to be save.
simil_method	specifies method to compute similarity between features. The value is passed to <code>quanteda.textstats::textstat_simil()</code> , "cosine" is used otherwise.
engine	select the engine to factorize x to generate word vectors. Choose from <code>RSpectra::svds()</code> , <code>irlba::irlba()</code> , <code>rsvd::rsvd()</code> , and <code>rsparse::GloVe()</code> .
auto_weight	automatically determine weights to approximate the polarity of terms to seed words. See details.
include_data	if TRUE, fitted model includes the dfm supplied as x.
group_data	if TRUE, apply <code>dfm_group(x)</code> before saving the dfm.
verbose	show messages if TRUE.
w	the size of word vectors. Used only when x is a fcm.
max_count	passed to <code>x_max</code> in <code>rsparse::GloVe\$new()</code> 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_1ss()` 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_1ss()` 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.

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	<i>Plot similarity between seed words</i>
----------------	---

Description

Plot similarity between seed words

Usage

```
textplot_simil(x)
```

Arguments

`x` fitted `textmodel_1ss` object.

textplot_terms	<i>Plot polarity scores of words</i>
----------------	--------------------------------------

Description

Plot polarity scores of words

Usage

```
textplot_terms(x, highlighted = NULL, max_highlighted = 50, max_words = 10000)
```

Arguments

x	a fitted textmodel_1ss object.
highlighted	quanteda::pattern to select words to highlight.
max_highlighted	the maximum number of words to highlight. When highlighted = NULL, words to highlight are randomly selected proportionally to $\text{polarity}^2 * \log(\text{frequency})$.
max_words	the maximum number of words to plot. Words are randomly sampled to keep the number below the limit.

textstat_context	<i>Identify context words using user-provided patterns</i>
------------------	--

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 <i>n</i> in the <i>n</i> -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 = <i>k</i> where <i>k</i> is the distance for which <i>k</i> or fewer skips are used to construct the <i>n</i> -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\(\)](#)

Index

- * **data**
 - data_textmodel_lass_russianprotests,
4
- as.seedwords, 2
- bootstrap_lass, 2
- char_context(textstat_context), 10
- coef.textmodel_lass, 3
- coefficients.textmodel_lass
(coef.textmodel_lass), 3
- data_dictionary_ideology, 4
- data_dictionary_sentiment, 4
- data_textmodel_lass_russianprotests, 4
- dictionary, 2
- irlba::irlba(), 8
- locfit(), 7
- loess(), 7
- lp(), 7
- predict.textmodel_lass, 5
- quanteda.textstats::textstat_simil(),
8
- quanteda::dfm(), 8
- quanteda::dfm_weight(), 8
- quanteda::fcm(), 8
- quanteda::pattern, 10
- quanteda::pattern(), 11
- quanteda::tokens(), 11
- quanteda::valuetype, 8
- quanteda::valuetype(), 11
- rsparse::GloVe(), 8
- RSpectra::svds(), 8
- rsvd::rsvd(), 8
- seedwords, 6
- smooth_lass, 6
- stats::optim(), 9
- textmodel_lass, 3, 7
- textplot_simil, 9
- textplot_terms, 10
- textstat_context, 10
- textstat_keyness(), 11
- tokens_select(), 11