Package ‘LSX’

April 19, 2021

Type Package
Title Model for Semisupervised Text Analysis Based on Word Embeddings
Date 2021-04-16
Version 0.9.9
LS 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 2021-04-19 09:30:02 UTC

R topics documented:

  as.seedwords ................................................................. 2
  cohesion ................................................................. 2
  data_dictionary_ideology ............................................. 3
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)

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

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


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

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`

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**


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

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

Arguments

- **x**: a dfm or fcm created by `quanteda::dfm()` or `quanteda::fcm()`
- **...**: additional argument passed to the SVD engine
- **seeds**: a character vector, named numeric vector or dictionary that contains seed words.
- **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".
textmodel_lss

**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**

choose SVD engine between RSpectra::svds(), irlba::irlba(), and rsparse::GloVe().

**include_data**

if TRUE, fitted model include the dfm supplied as x.

**verbose**

show messages if TRUE.

**w**

the size of word vectors. Only used when x is a fcm

### References


### Examples

```r
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, "politix\*-"), 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**

```r
textplot_simil(x, group = FALSE)
```

**Arguments**

- `x`: fitted `textmodel_lss` object
- `group`: if TRUE group seed words by seed patterns and show average similarity

---

**textplot_terms**  

*Plot polarity scores of words*

**Description**

Plot polarity scores of words

**Usage**

```r
textplot_terms(x, highlighted = NULL)
```

**Arguments**

- `x`: fitted `textmodel_lss` object
- `highlighted`: `quanteda::pattern` to specify words to highlight
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

```r
# @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

* data
  data_textmodel_lss_russianprotests, 3
  as.seedwords, 2
  char_context (textstat_context), 9
  char_keyness (textstat_context), 9
  cohesion, 2
  corpus, 4
  data_dictionary_ideology, 3
  data_dictionary_sentiment, 3
  data_textmodel_lss_russianprotests, 3
  diagnosys, 4
  dictionary, 2
  irlba::irlba(), 7
  locfit(), 5
  loess(), 5
  lp(), 5
  quanteda.textstats::textstat_simil(), 7
  quanteda::dfm(), 6
  quanteda::dfm_weight(), 6
  quanteda::fcm(), 6
  quanteda::pattern, 8
  quanteda::pattern(), 10
  quanteda::tokens(), 10
  quanteda::valuetype(), 10
  rsparse::GloVe(), 7
  RSpectra::svds(), 7
  seedwords, 4
  smooth_lss, 5
  textmodel_lss, 5
  textplot_simil, 8
  textplot_terms, 8
  textstat_context, 9
  textstat_keyness(), 10
  tokens_select(), 10