# Package ‘LSX’

July 20, 2021

**Type** Package

**Title** Model for Semisupervised Text Analysis Based on Word Embeddings

**Date** 2021-07-20

**Version** 1.0.0


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** 2021-07-20 05:00:06 UTC

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.seedwords</td>
<td>2</td>
</tr>
<tr>
<td>cohesion</td>
<td>3</td>
</tr>
<tr>
<td>data_dictionary_ideology</td>
<td>3</td>
</tr>
</tbody>
</table>
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**
```r
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**
```r
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**
```r
as.seedwords(data_dictionary_sentiment)
```
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
**predict.textmodel_lss**  
*Prediction method for textmodel_lss*

**Description**

Prediction method for textmodel_lss

**Usage**

```r
## 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**

```r
seedwords(type)
```

**Arguments**

- **type**: type of seed words currently only for sentiment (sentiment) or political ideology (ideology).
References


Examples

seedwords('sentiment')

smooth_lss

\textit{Smooth predicted LSS scores by local polynomial regression}

Description

Smooth predicted LSS scores by local polynomial regression

Usage

\begin{verbatim}
smooth_lss(
  x,
  lss_var = "fit",
  date_var = "date",
  span = 0.1,
  from = NULL,
  to = NULL,
  engine = c("loess", "locfit"),
  ...
)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} a \texttt{data.frame} containing LSS scores and dates
  \item \texttt{lss_var} the name of the column for LSS scores
  \item \texttt{date_var} the name of the columns for dates
  \item \texttt{span} determines the level of smoothing.
  \item \texttt{from} start of the time period
  \item \texttt{to} end of the time period
  \item \texttt{engine} specifies the function to smooth LSS scores: \texttt{loess()} or \texttt{locfit()}. The latter should be used when \texttt{n > 10000}.
  \item \texttt{...} extra arguments passed to \texttt{loess()} or \texttt{lp()}
\end{itemize}
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,  
max_count = 10,  
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 arguments passed to the underlying 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".

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()`.

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 coocurrence counts are ceiled to this threshold. It should be changed according to the size of the corpus. Used only 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, '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, 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
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

```r
# 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 \texttt{quanteda::tokens}.

pattern \texttt{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 \texttt{quanteda::valuetype()} for details.

case_insensitive if \texttt{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 \texttt{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 \textit{n} in the \textit{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 \textit{skip = k} where \textit{k} is the distance for which \textit{k} or fewer skips are used to construct the \textit{n}-gram. Thus a "4-skip-n-gram" defined as \textit{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 \texttt{textstat_keyness}().

p threshold for statistical significance of collocations.

See Also

\texttt{tokens.select()} and \texttt{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, 4
    as.seedwords, 2
  char_context (textstat_context), 10
  char_keyness (textstat_context), 10
  cohesion, 3
  corpus, 4
  data_dictionary_ideology, 3
  data_dictionary_sentiment, 3
  data_textmodel_lss_russianprotests, 4
  diagnosys, 4
  dictionary, 2
  irlba::irlba(), 8
  locfit(), 6
  loess(), 6
  lp(), 6
  predict.textmodel_lss, 5
  quanteda.textstats::textstat_simil(), 8
  quanteda::dfm(), 7, 8
  quanteda::dfm_weight(), 8
  quanteda::fcm(), 7, 8
  quanteda::pattern, 10
  quanteda::pattern(), 11
  quanteda::tokens(), 11
  quanteda::valuetype(), 11
  rsparse::GloVe(), 8
  RSpectra::svds(), 8
  rsvd::rsvd(), 8
  seedwords, 5
  smooth_lss, 6
  textmodel_lss, 7
  textplot_simil, 9
  textplot_terms, 9
  textstat_context, 10
  textstat_keyness(), 11
  tokens_select(), 11