Package ‘sentopics’

April 18, 2024

Type Package

Title Tools for Joint Sentiment and Topic Analysis of Textual Data

Version 0.7.3

Date 2024-04-17

Maintainer Olivier Delmarcelle <delmarcelle.olivier@gmail.com>

Description A framework that joins topic modeling and sentiment analysis of textual data. The package implements a fast Gibbs sampling estimation of Latent Dirichlet Allocation (Griffiths and Steyvers (2004) <doi:10.1073/pnas.0307752101>) and Joint Sentiment/Topic Model (Lin, He, Everson and Ruger (2012) <doi:10.1109/TKDE.2011.48>). It offers a variety of helpers and visualizations to analyze the result of topic modeling. The framework also allows enriching topic models with dates and externally computed sentiment measures. A flexible aggregation scheme enables the creation of time series of sentiment or topical proportions from the enriched topic models. Moreover, a novel method jointly aggregates topic proportions and sentiment measures to derive time series of topical sentiment.

License GPL (>= 3)

BugReports https://github.com/odelmarcelle/sentopics/issues

URL https://github.com/odelmarcelle/sentopics

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.4.6), methods, generics, quanteda (>= 3.2.0), data.table (>= 1.13.6), RcppHungarian

Suggests ggplot2, ggridges, plotly, RColorBrewer, xts, zoo, future, future.apply, progressr, progress, testthat, covr, stm, lda, topicmodels, seededLDA, keyATM, LDAvis, servr, textcat, stringr, sentometrics, spacyr, knitr, rmarkdown, webshot

LinkingTo Rcpp, RcppArmadillo, RcppProgress

RcppModules model_module

RoxygenNote 7.3.1
LazyData: true
VignetteBuilder: knitr
NeedsCompilation: yes

Author: Olivier Delmarcelle [aut, cre]  
        (https://orcid.org/0000-0003-4347-070X),  
        Samuel Borms [ctb] (https://orcid.org/0000-0001-9533-1870),  
        Chengua Lin [cph] (Original JST implementation),  
        Yulan He [cph] (Original JST implementation),  
        Jose Bernardo [cph] (Original JST implementation),  
        David Robinson [cph] (Implementation of reorder_within()),  
        Julia Silge [cph] (Implementation of reorder_within(),  
        (https://orcid.org/0000-0002-3671-836X)

Repository: CRAN
Date/Publication: 2024-04-18 13:12:38 UTC

R topics documented:

- sentopics-package
- as.LDA
- as.tokens.dfm
- chainsDistances
- chainsScores
- coherence
- compute_PicaultRenault_scores
- ECB_press_conferences
- ECB_press_conferences_tokens
- fit.sentopicmodel
- get_ECB_press_conferences
- get_ECB_speeches
- JST
- LDA
- LDavis
- LoughranMcDonald
- melt
- melt.sentopicmodel
- mergeTopics
- PicaultRenault
- PicaultRenault_data
- plot.multiChains
- plot.sentopicmodel
- print.sentopicmodel
- proportion_topics
- reset
- rJST
- sentiment_breakdown
- sentiment_series
**Description**

*sentopics* provides function to easily estimate a range of topic models and process their output. Particularly, it facilitates the integration of topic analysis with a time dimension through time-series generating functions. In addition, *sentopics* interacts with sentiment analysis to compute the sentiment conveyed by topics. Finally, the package implements a number of visualization helping interpreting the results of topic models.

**Usage**

Please refer to the vignettes for a comprehensive introduction to the package functions.

- **Basic usage**: Introduction to topic model estimation with *sentopics*
- **Topical time series**: Integrate topic analysis with sentiment analysis along a time dimension

For further details, you may browse the package documentation.

**Note**

Please cite the package in publications. Use `citation("sentopics")`.

**Author(s)**

**Maintainer**: Olivier Delmarcelle <delmarcelle.olivier@gmail.com> (ORCID)

Other contributors:

- Samuel Borms <samuel.borms@unine.ch> (ORCID) [contributor]
- Chengua Lin <chenghua.lin@abdn.ac.uk> (Original JST implementation) [copyright holder]
- Yulan He <yulan.he@warwick.ac.uk> (Original JST implementation) [copyright holder]
- Jose Bernardo (Original JST implementation) [copyright holder]
- David Robinson <admiral.david@gmail.com> (Implementation of reorder_within()) [copyright holder]
- Julia Silge <julia.silge@gmail.com> (ORCID) (Implementation of reorder_within()) [copyright holder]
See Also

Useful links:

- [https://github.com/odelmarcelle/sentopics](https://github.com/odelmarcelle/sentopics)
- Report bugs at [https://github.com/odelmarcelle/sentopics/issues](https://github.com/odelmarcelle/sentopics/issues)

---

**as.LDA**

Conversions from other packages to LDA

---

**Description**

These functions converts estimated models from other topic modeling packages to the format used by **sentopics**.

**Usage**

```r
as.LDA(x, ...)
```

## S3 method for class 'STM'

```r
as.LDA(x, docs, ...)
```

## S3 method for class 'LDA_Gibbs'

```r
as.LDA(x, docs, ...)
```

## S3 method for class 'LDA_VEM'

```r
as.LDA(x, docs, ...)
```

## S3 method for class 'textmodel_lda'

```r
as.LDA(x, ...)
```

```r
as.LDA_lda(list, docs, alpha, eta)
```

## S3 method for class 'keyATM_output'

```r
as.LDA(x, docs, ...)
```

**Arguments**

- `x` an estimated topic model from **stm**, **topicmodels** or **seededlda**.
- `...` arguments passed to other methods.
- `docs` for some objects, the documents used to initialize the model.
- `list` the list containing an estimated model from **lda**.
- `alpha` for **lda** models, the document-topic mixture hyperparameter. If missing, the hyperparameter will be set to 50/K.
- `eta` for **lda** models, the topic-word mixture hyperparameter. Other packages refer to this hyperparameter as **beta**. If missing, the hyperparameter will be set to 0.01.
Details

Some models do not store the topic assignment of each word (for example, estimated through variational inference). For these, the conversion is limited and some functionalities of sentopics will be disabled. The list of affected functions is subject to change and currently includes fit(), mergeTopics() and rJST.LDA().

Since models from the lda package are simply lists of outputs, the function as.LDA.lda() is not related to the other methods and should be applied directly on lists containing a model.

Value

A S3 list of class LDA, as if it was created and estimated using LDA() and fit().

Examples

```r
## stm
library("stm")
stm <- stm(poliblog5k.docs, poliblog5k.voc, K=25,
prevalence=~rating, data=poliblog5k.meta,
max.em.its=2, init.type="Spectral")
as.LDA(stm, docs = poliblog5k.docs)

## lda
library("lda")
data("cora.documents")
data("cora.vocab")
lda <- lda.collapsed.gibbs.sampler(cora.documents,
5, ## Num clusters
cora.vocab,
100, ## Num iterations
0.1,
0.1)
LDA <- as.LDA.lda(lda, docs = cora.documents, alpha = .1, eta = .1)

## topicmodels
data("AssociatedPress", package = "topicmodels")
lda <- topicmodels::LDA(AssociatedPress[1:20,],
control = list(alpha = 0.1), k = 2)
LDA <- as.LDA(lda, docs = AssociatedPress[1:20,])

## seededlda
library("seededlda")
lda <- textmodel_lda(dfm(ECB_press_conferences_tokens),
k = 6, max_iter = 100)
LDA <- as.LDA(lda)

## keyATM
library("keyATM")
data(keyATM_data_bills, package = "keyATM")
keyATM_docs <- keyATM_read(keyATM_data_bills$doc_dfm)
out <- keyATM(docs = keyATM_docs, model = "base")
```
As tokens dfm

Convert back a dfm to a tokens object

Description

Convert back a dfm to a tokens object

Usage

```r
## S3 method for class 'dfm'
as.tokens(
x, 
concatenator = NULL, 
tokens = NULL, 
ignore_list = NULL, 
case_insensitive = FALSE, 
padding = TRUE, 
...
)
```

Arguments

- `x` quanteda::dfm to be coerced
- `concatenator` only used for consistency with the generic
- `tokens` optionally, the tokens from which the dfm was created. Providing the initial tokens will ensure that the word order will be respected in the coerced object.
- `ignore_list` a character vector of words that should not be removed from the initial tokens object. Useful to avoid removing some lexicon word following the usage of quanteda::dfm_trim().
- `case_insensitive` only used when the tokens argument is provided. Default to FALSE. This function removes words in the initial tokens based on the remaining features in the dfm object. This check is case-sensitive by default, and can be relaxed by setting this argument to TRUE.
- `padding` if TRUE, leaves an empty string where the removed tokens previously existed. The use of padding is encouraged to improve the behavior of the coherence metrics (see coherence()) that rely on word positions.
- `...` unused

Value

a quanteda quanteda::tokens object.
chainsDistances

See Also
quanteda::as.tokens() quanteda::dfm()

Examples
library("quanteda")
dfm <- dfm(ECB_press_conferences_tokens, tolower = FALSE)
dfm <- dfm_trim(dfm, min_termfreq = 200)
as.tokens(dfm)
as.tokens(dfm, tokens = ECB_press_conferences_tokens)
as.tokens(dfm, tokens = ECB_press_conferences_tokens, padding = FALSE)

---

chainsDistances  Distances between topic models (chains)

Description
Computes the distance between different estimates of a topic model. Since the estimation of a topic model is random, the results may largely differ as the process is repeated. This function allows to compute the distance between distinct realizations of the estimation process. Estimates are referred to as chains.

Usage
chainsDistances(
  x,
  method = c("euclidean", "hellinger", "cosine", "minMax", "naiveEuclidean",
              "invariantEuclidean"),
  ...
)

Arguments
x       a valid multiChains object, obtained through the estimation of a topic model using fit() and the argument nChains greater than 1.
method   the method used to measure the distance between chains.
...      further arguments passed to internal distance functions.

Details
The method argument determines how are computed distance.

- euclidean finds the pairs of topics that minimizes and returns the total Euclidean distance.
- hellinger does the same but based on the Hellinger distance.
- cosine does the same but based on the Cosine distance.
- minMax computes the maximum distance among the best pairs of distances. Inspired by the minimum-matching distance from Tang et al. (2014).
chainsScores

- naiveEuclidean computes the Euclidean distance without searching for the best pairs of topics.
- invariantEuclidean computes the best pairs of topics for all allowed permutations of topic indices. For JST and reversed-JST models, the two-levels hierarchy of document-sentiment-topic leads some permutations of indices to represent a drastically different outcome. This setting restricts the set of permutations to the ones that do not change the interpretation of the model. Equivalent to euclidean for LDA models.

Value

A matrix of distance between the elements of x

Author(s)

Olivier Delmarcelle

References


See Also

plot.multiChains() chainsScores()

Examples

```r
model <- LDA(ECB_press_conferences_tokens)
model <- fit(model, 10, nChains = 5)
chainsDistances(model)
```

<table>
<thead>
<tr>
<th>chainsScores</th>
<th>Compute scores of topic models (chains)</th>
</tr>
</thead>
</table>

Description

Compute various scores (log likelihood, coherence) for a list of topic models.

Usage

```r
chainsScores(x, window = 110, nWords = 10)
```
coherence

Arguments

- `x`: a valid `multiChains` object, obtained through the estimation of a topic model using `fit()` and the argument `nChains` greater than 1.
- `window`: optional. If `NULL`, use the default window for each coherence metric (10 for C_NPMI and 110 for C_V). It is possible to override these default windows by providing an integer or "boolean" to this argument, determining a new window size for all measures.
- `nWords`: the number of words used to compute coherence. See `coherence()`.

Value

A `data.table` with some statistics about each chain. For the coherence metrics, the value shown is the mean coherence across all topics of a chain.

Parallelism

When `nChains > 1`, the function can take advantage of `future.apply::future_lapply` (if installed) to spread the computation over multiple processes. This requires the specification of a parallel strategy using `future::plan()`. See the examples below.

See Also

`chainsDistances()` `coherence()`

Examples

```r
model <- LDA(ECB_press_conferences_tokens[1:10])
model <- fit(model, 10, nChains = 5)
chainsScores(model, window = 5)
chainsScores(model, window = "boolean")

# -- Parallel computation --
require(future.apply)
future::plan("multisession", workers = 2) # Set up 2 workers
chainsScores(model, window = "boolean")

future::plan("sequential") # Shut down workers
```

<table>
<thead>
<tr>
<th>coherence</th>
<th>Coherence of estimated topics</th>
</tr>
</thead>
</table>

Description

Computes various coherence based metrics for topic models. It assesses the quality of estimated topics based on co-occurrences of words. For best results, consider cleaning the initial tokens object with `padding = TRUE`.
coherence

Usage

coherence(
  x,
  nWords = 10,
  method = c("C_NPMI", "C_V"),
  window = NULL,
  NPMIs = NULL
)

Arguments

  x              a model created from the LDA(), JST() or rJST() function and estimated with fit()

  nWords         the number of words in each topic used for evaluation.

  method         the coherence method used.

  window         optional. If NULL, use the default window for each coherence metric (10 for C_NPMI and 110 for C_V). It is possible to override these default windows by providing an integer or "boolean" to this argument, determining a new window size for all measures. No effect is the NPMIs argument is also provided.

  NPMIs          optional NPMI matrix. If provided, skip the computation of NPMI between words, substantially decreasing computing time.

Details

Currently, only C_NPMI and C_V are documented. The implementation follows Röder & al. (2015). For C_NPMI, the sliding window is 10 whereas it is 110 for C_V.

Value

A vector or matrix containing the coherence score of each topic.

Author(s)

Olivier Delmarcelle

References

**compute_PicaultRenault_scores**

*Compute scores using the Picault-Renault lexicon*

**Description**

Computes Monetary Policy and Economic Condition scores using the Picault-Renault lexicon for central bank communication.

**Usage**

```r
compute_PicaultRenault_scores(x, min_ngram = 2, return_dfm = FALSE)
```

**Arguments**

- `x`: a `quanteda::corpus` object.
- `min_ngram`: the minimum length of n-grams considered in the computation
- `return_dfm`: if TRUE, returns the scaled word-per-document score under two `dfm`, on for the Monetary Policy and one for the Economic Condition categories. If FALSE, returns the sum of all word scores per document.

**Details**

The computation is done on a per-document basis, such as each document is scored with a value between -1 and 1. This is relevant to the computation of the denominator of the score.

It is possible to compute the score for paragraphs and sentences for a `quanteda::corpus` segmented using `quanteda::corpus_reshape`. Segmenting a corpus using `quanteda`’s helpers retain track to which document each paragraph/sentence belong. However, in that case, it is possible that paragraphs or sentences are scored outside the (-1,1) interval. In any case, the of the paragraph/sentences scores averaged over documents will be contained in the (-1,1) interval.

**Value**

A matrix with two columns, indicating respectively the MP (Monetary Policy) and EC (Economic Condition) scores of each document.

**References**


**See Also**

PicaultRenault
Examples

```r
# on documents
docs <- quanteda::corpus_reshape(ECB_press_conferences, "documents")
compute_PicaultRenault_scores(docs)

# on paragraphs
compute_PicaultRenault_scores(ECB_press_conferences)
```

---

**ECB_press_conferences**  
*Corpus of press conferences from the European Central Bank*

---

**Description**

A corpus of 260 ECB press conference, split into 4224 paragraphs. The corpus contains a number of `docvars` indicating the date of the press conference and a measured sentiment based on the Loughran-McDonald lexicon.

**Usage**

`ECB_press_conferences`

**Format**

A `quanteda::corpus` object.

**Source**


**See Also**

`ECB_press_conferences_tokens`

**Examples**

`docvars(ECB_press_conferences)`
Description

The pre-processed and tokenized version of the ECB_press_conferences corpus of press conferences. The processing involved the following steps:

- Subset paragraphs shorter than 10 words
- Removal of stop words
- Part-of-speech tagging, following which only nouns, proper nouns and adjective were retained.
- Detection and merging of frequent compound words
- Frequency-based cleaning of rare and very common words

Usage

ECB_press_conferences_tokens

Format

A quanteda::tokens object.

Source


See Also

ECB_press_conferences

Examples

LDA(ECB_press_conferences_tokens)
Description

This function is used to estimate a topic model created by \texttt{LDA()}, \texttt{JST()} or \texttt{rJST()}. In essence, this function iterates a Gibbs sampler MCMC.

Usage

```r
## S3 method for class 'sentopicmodel'
fit(
  object,
  iterations = 100,
  nChains = 1,
  displayProgress = TRUE,
  computeLikelihood = TRUE,
  seed = NULL,
  ...
)

## S3 method for class 'multiChains'
fit(
  object,
  iterations = 100,
  nChains = NULL,
  displayProgress = TRUE,
  computeLikelihood = TRUE,
  seed = NULL,
  ...
)
```

Arguments

- **object**: a model created with the \texttt{LDA()}, \texttt{JST()} or \texttt{rJST()} function.
- **iterations**: the number of iterations by which the model should be fitted.
- **nChains**: if set above 1, the model will be fitted multiple times from various starting positions. Latent variables will be re-initialized if \texttt{object} has not been fitted before.
- **displayProgress**: if \text{TRUE}, a progress bar will be displayed indicating the progress of the computation. When \texttt{nChains} is greater than 1, this requires the package \texttt{progressr} and optionally \texttt{progress}.
- **computeLikelihood**: if set to \text{FALSE}, does not compute the likelihood at each iteration. This can slightly decrease the computing time.
- **seed**: for reproducibility, a seed can be provided.
- **...**: arguments passed to other methods. Not used.
Value

A `sentopicmodel` object of the relevant model class if `nChains` is unspecified or equal to 1. A `multiChains` object if `nChains` is greater than 1.

Parallelism

When `nChains > 1`, the function can take advantage of `future::future_lapply` (if installed) to spread the computation over multiple processes. This requires the specification of a parallel strategy using `future::plan()`. See the examples below.

See Also

`LDA()`, `JST()`, `rJST()`, `reset()`

Examples

```r
model <- rJST(ECB_press_conferences_tokens)
fit(model, 10)

# -- Parallel computation --
require(future.apply)
future::plan("multisession", workers = 2) # Set up 2 workers
fit(model, 10, nChains = 2)

future::plan("sequential") # Shut down workers
```

get_ECB_press_conferences

`Download press conferences from the European Central Bank`

Description

This helper function automatically retrieve the full data set of press conferences made available by the ECB. It implements a number of pre-processing steps used to remove the Q&A section from the text.

Usage

```r
get_ECB_press_conferences(
    years = 1998:2021,
    language = "en",
    data.table = TRUE
)
```
get_ECB_speeches

Arguments

years
language
data.table

the years for which press conferences should be retrieved
the language in which press conferences should be retrieved
if TRUE, returns a data.table. Otherwise, return a list in which each element is a press conference.

Value

Depending on the arguments, returns either a data.frame or a quanteda::tokens object containing press conferences of the ECB.

get_ECB_speeches

Download and pre-process speeches from the European Central Bank

Description

This helper function automatically retrieve the full data set of speeches made available by the ECB. In addition, it implements a number of pre-processing steps that may be turned on or off as needed.

Usage

get_ECB_speeches(

  filter_english = TRUE,
  clean_footnotes = TRUE,
  compute_sentiment = TRUE,
  tokenize_w_POS = FALSE
)

Arguments

filter_english
if TRUE, attempts to select English speeches only using textcat::textcat().
clean_footnotes
if TRUE, attempts to clean footnotes from speeches texts using some regex patterns.
compute_sentiment
if TRUE, computes the sentiment of each speech using sentometrics::compute_sentiment() with the the Loughran & McDonald lexicon.
tokenize_w_POS
if TRUE, tokenizes and apply Part-Of-Speech tagging with spacyr::spacy_parse(). Nouns, adjectives and proper nouns are then extracted from the parsed speeches to form a tokens object.

Value

Depending on the arguments, returns either a data.frame or a quanteda::tokens object containing speeches of the ECB.
Create a Joint Sentiment/Topic model

Description

This function initialize a Joint Sentiment/Topic model.

Usage

```r
JST(
  x,
  lexicon = NULL,
  S = 3,
  K = 5,
  gamma = 1,
  alpha = 5,
  beta = 0.01,
  gammaCycle = 0,
  alphaCycle = 0
)
```

Arguments

- `x`: tokens object containing the texts. A coercion will be attempted if `x` is not a tokens.
- `lexicon`: a quanteda dictionary with positive and negative categories
- `S`: the number of sentiments
- `K`: the number of topics
- `gamma`: the hyperparameter of sentiment-document distribution
- `alpha`: the hyperparameter of topic-document distribution
- `beta`: the hyperparameter of vocabulary distribution
- `gammaCycle`: integer specifying the cycle size between two updates of the hyperparameter `alpha`
- `alphaCycle`: integer specifying the cycle size between two updates of the hyperparameter `alpha`

Details

The `rJST.LDA` methods enable the transition from a previously estimated LDA model to a sentiment-aware `rJST` model. The function retains the previously estimated topics and randomly assigns sentiment to every word of the corpus. The new model will retain the iteration count of the initial LDA model.
Value

An S3 list containing the model parameter and the estimated mixture. This object corresponds to a Gibbs sampler estimator with zero iterations. The MCMC can be iterated using the `fit()` function.

- `tokens` is the tokens object used to create the model
- `vocabulary` contains the set of words of the corpus
- `it` tracks the number of Gibbs sampling iterations
- `za` is the list of topic assignment, aligned to the `tokens` object with padding removed
- `logLikelihood` returns the measured log-likelihood at each iteration, with a breakdown of the likelihood into hierarchical components as attribute

The `topWords()` function easily extract the most probables words of each topic/sentiment.

Author(s)

Olivier Delmarcelle

References


See Also

Fitting a model: `fit()`, extracting top words: `topWords()`

Other topic models: `LDA()`, `rJST()`, `sentopicmodel()`

Examples

```r
# creating a JST model
JST(ECB_press_conferences_tokens)

# estimating a JST model including a lexicon
jst <- JST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
jst <- fit(jst, 100)
```
Create a Latent Dirichlet Allocation model

Description

This function initialize a Latent Dirichlet Allocation model.

Usage

LDA(x, K = 5, alpha = 1, beta = 0.01)

Arguments

- x : tokens object containing the texts. A coercion will be attempted if x is not a tokens.
- K : the number of topics
- alpha : the hyperparameter of topic-document distribution
- beta : the hyperparameter of vocabulary distribution

Details

The rJST.LDA methods enable the transition from a previously estimated LDA model to a sentiment-aware rJST model. The function retains the previously estimated topics and randomly assigns sentiment to every word of the corpus. The new model will retain the iteration count of the initial LDA model.

Value

An S3 list containing the model parameter and the estimated mixture. This object corresponds to a Gibbs sampler estimator with zero iterations. The MCMC can be iterated using the fit() function.

- tokens is the tokens object used to create the model
- vocabulary contains the set of words of the corpus
- it tracks the number of Gibbs sampling iterations
- za is the list of topic assignment, aligned to the tokens object with padding removed
- logLikelihood returns the measured log-likelihood at each iteration, with a breakdown of the likelihood into hierarchical components as attribute

The topWords() function easily extract the most probables words of each topic/sentiment.

Author(s)

Olivier Delmarcelle
References


See Also

Fitting a model: fit(), extracting top words: topWords()
Other topic models: JST(), rJST(), sentopicmodel()

Examples

# creating a model
LDA(ECB_press_conferences_tokens, K = 5, alpha = 0.1, beta = 0.01)

# estimating an LDA model
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)

LDAvis

Visualize a LDA model using LDAvis

Description

This function call LDAvis to create a dynamic visualization of an estimated topic model.

Usage

LDAvis(x, ...)

Arguments

x an LDA model
...

Details

The CRAN release of LDAvis does not support UTF-8 characters and automatically reorder topics. To solve these two issues, please install the development version of LDAvis from github (devtools::install_github("cpsievert/LDAvis")).

Value

Nothing, called for its side effects.

See Also

plot.sentopicmodel()
LoughranMcDonald

Examples

```r
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
LDAvis(lda)
```

---

LoughranMcDonald  \textit{Loughran-McDonald lexicon}

---

Description

The Loughran-McDonald lexicon for financial texts adapted for usage in \textit{sentopics}. The lexicon is enhanced with two list of valence-shifting words.

Usage

LoughranMcDonald

Format

A \texttt{quanteda::dictionary} containing two polarity categories (negative and positive) and two valence-shifting categories (negator and amplifier).

Source

https://sraf.nd.edu/loughranmcdonald-master-dictionary/ for the lexicon and \texttt{lexicon::hash_valence_shifters} for the valence shifters.

References


See Also

\texttt{JST()}, \texttt{rJST()}

Examples

```r
JST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
```
melt

Replacement generic for data.table::melt()

Description
As of the CRAN release of the 1.14.8 version of data.table, the data.table::melt() function is not a generic. This function aims to temporary provide a generic to this function, so that melt.sentopicmodel() can be effectively dispatched when used. Expect this function to disappear shortly after the release of data.table 1.14.9.

Usage
melt(data, ...)

Arguments
  data an object to melt
  ... arguments passed to other methods

Value
An unkeyed data.table containing the molten data.

See Also
data.table::melt(), melt.sentopicmodel()

melt.sentopicmodel Melt for sentopicmodels

Description
This function extracts the estimated document mixtures from a topic model and returns them in a long data.table format.

Usage
## S3 method for class 'sentopicmodel'
melt(data, ..., include_docvars = FALSE)

Arguments
  data a model created from the LDA(), JST() or rJST() function and estimated with fit()
  ... not used
  include_docvars if TRUE, the melted result will also include the docvars stored in the tokens object provided at model initialization
mergeTopics

Value

A data.table in the long format, where each line is the estimated proportion of a single topic/sentiment for a document. For JST and rJST models, the probability is also decomposed into 'L1' and 'L2' layers, representing the probability at each layer of the topic-sentiment hierarchy.

Author(s)

Olivier Delmarcelle

See Also

topWords() for extracting representative words, data.table::melt() and data.table::dcast()

Examples

# only returns topic proportion for LDA models
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 10)
melt(lda)

# includes sentiment for JST and rJST models
jst <- JST(ECB_press_conferences_tokens)
jst <- fit(jst, 10)
melt(jst)

mergeTopics

Merge topics into fewer themes

Description

This operation is especially useful for the analysis of the model’s output, by grouping together topics that share a common theme.

Usage

mergeTopics(x, merging_list)

Arguments

x a LDA() or rJST() model.
merging_list a list where each element is an integer vector containing the indices of topics to be merged. If named, the list’s names become the label of the aggregated themes.
Details

Topics are aggregated at the word assignment level. New document-topic and topic-word probabilities are derived from the aggregated assignments.

Note that the output of this function does not constitute an estimated topic model, but merely an aggregation to ease the analysis. It is not advised to use `fit()` on the merged topic model as it will radically affect the content and proportions of the new themes.

Value

A `LDA()` or `rJST()` model with the merged topics.

See Also

`sentopics_labels`

Examples

```r
lda <- LDA(ECB_press_conferences_tokens, K = 5)
lda <- fit(lda, 100)
merging_list <- list(
  c(1,5),
  2:4
)
mergeTopics(lda, merging_list)

# also possible with a named list
merging_list2 <- list(
  mytheme_1 = c(1,5),
  mytheme_2 = 2:4
)
merged <- mergeTopics(lda, merging_list2)
sentopics_labels(merged)

# implemented for rJST
rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
rjst <- fit(rjst, 100)
mergeTopics(rjst, merging_list2)
```

Description

The Picault-Renault lexicon, specialized in the analysis of central bank communication. The lexicon identifies a large number of n-grams and gives their probability to belong to six categories:

- Monetary Policy - accommodative
- Monetary Policy - neutral

---

**PicaultRenault**

**Picault-Renault lexicon**

---

PicaultRenault
PicaultRenault_data

- Monetary Policy - restrictive
- Economic Condition - negative
- Economic Condition - neutral
- Economic Condition - positive

Usage

PicaultRenault

Format

A data.table object.

Source

http://www.cbcomindex.com/lexicon.php

References


See Also

compute_PicaultRenault_scores()

Examples

head(PicaultRenault)

---

PicaultRenault_data  Regression dataset based on Picault & Renault (2017)

Description

A regression dataset built to partially replicate the result of Picault & Renault. This dataset contains, for each press conference published after 2000:

- The Main Refinancing Rate (MRR) of the ECB set following the press conference
- The change in the MRR following the press conference
- The change in the MRR observed at the previous press conference
- The Bloomberg consensus on the announced MRR
- The Surprise brought by the announcement, computed as the Bloomberg consensus minus the MRR following the conference
- The EURO STOXX 50 return on the day of the press conference
- The EURO STOXX 50 return on the day preceding the announcement
plot.multiChains

**Usage**

PicaultRenault_data

**Format**

An `xts::xts` object.

**Source**

The data was manually prepared by the author of this package.

**References**


**Examples**

```
head(PicaultRenault_data)
```

---

**plot.multiChains**  
*Plot the distances between topic models (chains)*

**Description**

Plot the results of `chainsDistance(x)` using multidimensional scaling. See `chainsDistances()` for details on the distance computation and `stats::cmdscale()` for the implementation of the multidimensional scaling.

**Usage**

```r
## S3 method for class 'multiChains'
plot(
  x,
  ..., 
  method = c("euclidean", "hellinger", "cosine", "minMax", "naiveEuclidean", 
    "invariantEuclidean")
)
```

**Arguments**

- `x` a valid `multiChains` object, obtained through the estimation of a topic model using `fit()` and the argument `nChains` greater than 1.
- `...` not used
- `method` the method used to measure the distance between chains.
Value

Invisibly, the coordinates of each topic model resulting from the multidimensional scaling.

See Also

chainsDistances() cmdscale()

Examples

```r
models <- LDA(ECB_press_conferences_tokens)
models <- fit(models, 10, nChains = 5)
plot(models)
```

plot.sentopicmodel

Plot a topic model using Plotly

Description

Summarize and plot a sentopics model using a sunburst chart from the plotly::plotly library.

Usage

```r
## S3 method for class 'sentopicmodel'
plot(x, nWords = 15, layers = 3, sort = FALSE, ...)
```

Arguments

- `x` a model created from the `LDA()`, `JST()` or `rJST()` function and estimated with `fit()`
- `nWords` the number of words per topic/sentiment to display in the outer layer of the plot
- `layers` specifies the number of layers for the sunburst chart. This will restrict the output to the layers uppermost levels of the chart. For example, setting `layers = 1` will only display the top level of the hierarchy (topics for an LDA model).
- `sort` if TRUE, sorts the plotted topics in a decreasing frequency.
- `...` not used

Value

A plotly sunburst chart.

See Also

`topWords() LDAvis()`
Examples

```r
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
plot(lda, nWords = 5)

# only displays the topic proportions
plot(lda, layers = 1)
```

print.sentopicmodel

Print method for sentopics models

Description

Print methods for sentopics models. Once per session (or forced by using extended = TRUE), it lists the most important function related to sentopics models.

Usage

```r
## S3 method for class 'sentopicmodel'
print(x, extended = FALSE, ...)

## S3 method for class 'rJST'
print(x, extended = FALSE, ...)

## S3 method for class 'LDA'
print(x, extended = FALSE, ...)

## S3 method for class 'JST'
print(x, extended = FALSE, ...)
```

Arguments

- `x` the model to be printed
- `extended` if TRUE, extends the print to include some helpful related functions. Automatically displayed once per session.
- `...` not used

Value

No return value, called for side effects (printing).
proportion_topics  Compute the topic or sentiment proportion time series

Description

Aggregate the topical or sentiment proportions at the document level into time series.

Usage

proportion_topics(
  x,
  period = c("year", "quarter", "month", "day", "identity"),
  rolling_window = 1,
  complete = TRUE,
  plot = c(FALSE, TRUE, "silent"),
  plot_ridgelines = TRUE,
  as.xts = TRUE,
  ...
)

plot_proportion_topics(
  x,
  period = c("year", "quarter", "month", "day"),
  rolling_window = 1,
  complete = TRUE,
  plot_ridgelines = TRUE,
  ...
)

Arguments

x  a LDA(), JST() or rJST() model populated with internal dates and/or internal sentiment.

period  the sampling period within which the sentiment of documents will be averaged. period = "identity" is a special case that will return document-level variables before the aggregation happens. Useful to rapidly compute topical sentiment at the document level.

rolling_window  if greater than 1, determines the rolling window to compute a moving average of sentiment. The rolling window is based on the period unit and rely on actual dates (i.e, is not affected by unequally spaced data points).

complete  if FALSE, only compute proportions at the upper level of the topic model hierarchy (topics for rJST and sentiment for JST). No effect on LDA models.

plot  if TRUE, prints a plot of the time series and attaches it as an attribute to the returned object. If 'silent', do not print the plot but still attaches it as an attribute.
plot_ridgelines

if TRUE, time series are plotted as ridgelines. Requires ggridges package installed. If FALSE, the plot will use only standard ggplot2 functions. If the argument is missing and the package ggridges is not installed, this will quietly switch to a ggplot2 output.

as.xts

if TRUE, returns an xts::xts object. Otherwise, returns a data.frame.

... other arguments passed on to zoo::rollapply() or mean() and sd().

Value

A time series of proportions, stored as an xts::xts object or as a data.frame.

See Also

sentopics_sentiment sentopics_date

Other series functions: sentiment_breakdown(), sentiment_series(), sentiment_topics()

Examples

lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
proportion_topics(lda)

# plot shortcut
plot_proportion_topics(lda, period = "month", rolling_window = 3)
# with or without ridgelines
plot_proportion_topics(lda, period = "month", plot_ridgelines = FALSE)

# also available for rJST and JST models
jst <- JST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
jst <- fit(jst, 100)
# including both layers
proportion_topics(jst)
# or not
proportion_topics(jst, complete = FALSE)

reset

Re-initialize a topic model

Description

This function is used re-initialize a topic model, as if it was created from LDA(), JST() or another model. The re-initialized model retains its original parameter specification.

Usage

reset(object)
Arguments

object a model created from the \texttt{LDA()}, \texttt{JST()} or \texttt{rJST()} function and estimated with \texttt{fit()}

Value

a sentopicmodel of the relevant model class, with the iteration count reset to zero and re-initialized assignment of latent variables.

Author(s)

Olivier Delmarcelle

See Also

\texttt{fit()}

Examples

\begin{verbatim}
model <- LDA(ECB_press_conferences_tokens)
model <- fit(model, 10)
reset(model)
\end{verbatim}

\section*{Description}

This function initialize a Reversed Joint Sentiment/Topic model.

Usage

\begin{verbatim}
rJST(x, ...)  
\end{verbatim}

## Default S3 method:
rJST(
x,  
lexicon = NULL,  
K = 5,  
S = 3,  
alpha = 1,  
gamma = 5,  
beta = 0.01,  
alphaCycle = 0,  
gammaCycle = 0,  
...  
)
## S3 method for class 'LDA'

`rJST(x, lexicon = NULL, S = 3, gamma = 5, ...)`

### Arguments

- `x` tokens object containing the texts. A coercion will be attempted if `x` is not a tokens.
- `...` not used
- `lexicon` a quanteda dictionary with positive and negative categories
- `K` the number of topics
- `S` the number of sentiments
- `alpha` the hyperparameter of topic-document distribution
- `gamma` the hyperparameter of sentiment-document distribution
- `beta` the hyperparameter of vocabulary distribution
- `alphaCycle` integer specifying the cycle size between two updates of the hyperparameter `alpha`
- `gammaCycle` integer specifying the cycle size between two updates of the hyperparameter `alpha`

### Details

The `rJST.LDA` methods enable the transition from a previously estimated LDA model to a sentiment-aware rJST model. The function retains the previously estimated topics and randomly assigns sentiment to every word of the corpus. The new model will retain the iteration count of the initial LDA model.

### Value

An S3 list containing the model parameter and the estimated mixture. This object corresponds to a Gibbs sampler estimator with zero iterations. The MCMC can be iterated using the `fit()` function.

- `tokens` is the tokens object used to create the model
- `vocabulary` contains the set of words of the corpus
- `it` tracks the number of Gibbs sampling iterations
- `za` is the list of topic assignment, aligned to the tokens object with padding removed
- `logLikelihood` returns the measured log-likelihood at each iteration, with a breakdown of the likelihood into hierarchical components as attribute

The `topWords()` function easily extract the most probables words of each topic/sentiment.

### Author(s)

Olivier Delmarcelle
sentiment_breakdown

References


See Also

Fitting a model: fit(), extracting top words: topWords()
Other topic models: JST(), LDA(), sentopicmodel()

Examples

# simple rJST model
rJST(ECB_press_conferences_tokens)

# estimating a rJST model including lexicon
rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
rjst <- fit(rjst, 100)

# from an LDA model:
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)

# creating a rJST model out of it
rjst <- rJST(lda, lexicon = LoughranMcDonald)
# topic proportions remain identical
identical(lda$theta, rjst$theta)
# model should be iterated to estimate sentiment proportions
rjst <- fit(rjst, 100)

sentiment_breakdown

Breakdown the sentiment into topical components

Description

Break down the sentiment series obtained with sentiment_series() into topical components. Sentiment is broken down at the document level using estimated topic proportions, then processed to create a time series and its components.

Usage

sentiment_breakdown(
  x,
  period = c("year", "quarter", "month", "day", "identity"),
  rolling_window = 1,
  scale = TRUE,
  scaling_period = c("1900-01-01", "2099-12-31"),
)
plot = c(FALSE, TRUE, "silent"),
as.xts = TRUE,
...
)

plot_sentiment_breakdown(
  x,
  period = c("year", "quarter", "month", "day"),
  rolling_window = 1,
  scale = TRUE,
  scaling_period = c("1900-01-01", "2099-12-31"),
  ...
)

Arguments

  x a LDA() or rJST() model populated with internal dates and/or internal sentiment.

  period the sampling period within which the sentiment of documents will be averaged. period = "identity" is a special case that will return document-level variables before the aggregation happens. Useful to rapidly compute topical sentiment at the document level.

  rolling_window if greater than 1, determines the rolling window to compute a moving average of sentiment. The rolling window is based on the period unit and rely on actual dates (i.e., is not affected by unequally spaced data points).

  scale if TRUE, the resulting time series will be scaled to a mean of zero and a standard deviation of 1. This argument also has the side effect of attaching scaled sentiment values as docvars to the input object with the _scaled suffix.

  scaling_period the date range over which the scaling should be applied. Particularly useful to normalize only the beginning of the time series.

  plot if TRUE, prints a plot of the time series and attaches it as an attribute to the returned object. If ‘silent’, do not print the plot but still attaches it as an attribute.

  as.xts if TRUE, returns an xts::xts object. Otherwise, returns a data.frame.

  ... other arguments passed on to zoo::rollapply() or mean() and sd().

Details

The sentiment is broken down at the sentiment level assuming the following composition:

\[ s = \sum_{i=1}^{K} s_i \times \theta_i \]

where \( s_i \) is the sentiment of topic \( i \) and \( \theta_i \) the proportion of topic \( i \) in a given document. For an LDA model, the sentiment of each topic is considered equal to the document sentiment (i.e. \( s_i = s \forall i \in K \)). The topical sentiment attention, defined by \( s*_{i} = s_i \times \theta_i \) represent the effective sentiment conveyed by a topic in a document. The topical sentiment attention of all documents in a period are averaged to compute the breakdown of the sentiment time series.
sentiment_series

Value

A time series of sentiment, stored as an xts::xts object or as a data.frame.

See Also

sentopics_sentiment sentopics_date

Other series functions: proportion_topics(), sentiment_series(), sentiment_topics()

Examples

lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
sentiment_breakdown(lda)

# plot shortcut
plot_sentiment_breakdown(lda)

# also available for rJST models (with topic-level sentiment)
rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
rjst <- fit(rjst, 100)
sentopics_sentiment(rjst, override = TRUE)
plot_sentiment_breakdown(rjst)

sentiment_series(x, period = c("year", "quarter", "month", "day"), rolling_window = 1, scale = TRUE, scaling_period = c("1900-01-01", "2099-12-31"), as.xts = TRUE, ...)

Description

Compute a sentiment time series based on the internal sentiment and dates of a sentopicmodel. The time series computation supports multiple sampling period and optionally allow computing a moving average.

Usage

sentiment_series(x, period = c("year", "quarter", "month", "day"), rolling_window = 1, scale = TRUE, scaling_period = c("1900-01-01", "2099-12-31"), as.xts = TRUE, ...)


Arguments

- **x**: a LDA(), JST() or rJST() model populated with internal dates and/or internal sentiment.
- **period**: the sampling period within which the sentiment of documents will be averaged. period = "identity" is a special case that will return document-level variables before the aggregation happens. Useful to rapidly compute topical sentiment at the document level.
- **rolling_window**: if greater than 1, determines the rolling window to compute a moving average of sentiment. The rolling window is based on the period unit and rely on actual dates (i.e., is not affected by unequally spaced data points).
- **scale**: if TRUE, the resulting time series will be scaled to a mean of zero and a standard deviation of 1. This argument also has the side effect of attaching scaled sentiment values as docvars to the input object with the _scaled suffix.
- **scaling_period**: the date range over which the scaling should be applied. Particularly useful to normalize only the beginning of the time series.
- **as.xts**: if TRUE, returns an xts::xts object. Otherwise, returns a data.frame.
- **...**: other arguments passed on to zoo::rollapply() or mean() and sd().

Value

A time series of sentiment, stored as an xts::xts or data.frame.

See Also

sentopics_sentiment, sentopics_date

Other series functions: proportion_topics(), sentiment_breakdown(), sentiment_topics()

Examples

```r
lda <- LDA(ECB_press_conferences_tokens)
series <- sentiment_series(lda, period = "month")

# JST and rJST models can use computed sentiment from the sentiment layer, # but the model must be estimated first.
# rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
sentiment_series(rjst)

sentopics_sentiment(rjst) <- NULL  # remove existing sentiment
rjst <- fit(rjst, 10)  # estimating the model is then needed
sentiment_series(rjst)

# note the presence of both raw and scaled sentiment values # in the initial object
sentopics_sentiment(lda)
sentopics_sentiment(rjst)
```
**sentiment_topics**

*Compute time series of topical sentiments*

**Description**

Derive topical time series of sentiment from a LDA() or rJST() model. The time series are created by leveraging on estimated topic proportions and internal sentiment (for LDA models) or topical sentiment (for rJST models).

**Usage**

```r
sentiment_topics(
  x,
  period = c("year", "quarter", "month", "day", "identity"),
  rolling_window = 1,
  scale = TRUE,
  scaling_period = c("1900-01-01", "2099-12-31"),
  plot = c(FALSE, TRUE, "silent"),
  plot_ridgelines = TRUE,
  as.xts = TRUE,
  ...
)
```

```r
plot_sentiment_topics(
  x,
  period = c("year", "quarter", "month", "day"),
  rolling_window = 1,
  scale = TRUE,
  scaling_period = c("1900-01-01", "2099-12-31"),
  plot_ridgelines = TRUE,
  ...
)
```

**Arguments**

- **x** a LDA() or rJST() model populated with internal dates and/or internal sentiment.
- **period** the sampling period within which the sentiment of documents will be averaged. period = "identity" is a special case that will return document-level variables before the aggregation happens. Useful to rapidly compute topical sentiment at the document level.
- **rolling_window** if greater than 1, determines the rolling window to compute a moving average of sentiment. The rolling window is based on the period unit and rely on actual dates (i.e., is not affected by unequally spaced data points).
- **scale** if TRUE, the resulting time series will be scaled to a mean of zero and a standard deviation of 1. This argument also has the side effect of attaching scaled sentiment values as docvars to the input object with the _scaled suffix.
sentiment_topics

- **scaling_period**: the date range over which the scaling should be applied. Particularly useful to normalize only the beginning of the time series.
- **plot**: if TRUE, prints a plot of the time series and attaches it as an attribute to the returned object. If 'silent', do not print the plot but still attaches it as an attribute.
- **plot_ridgelines**: if TRUE, time series are plotted as ridgelines. Requires ggridges package installed. If FALSE, the plot will use only standards ggplot2 functions. If the argument is missing and the package ggridges is not installed, this will quietly switch to a ggplot2 output.
- **as.xts**: if TRUE, returns an xts::xts object. Otherwise, returns a data.frame.
- **...**: other arguments passed on to zoo::rollapply() or mean() and sd().

**Details**

A topical sentiment is computed at the document level for each topic. For an LDA model, the sentiment of each topic is considered equal to the document sentiment (i.e. $s_i = s_{\forall i \in K}$). For a rJST model, these result from the proportions in the sentiment layer under each topic. To compute the topical time series, the topical sentiment of all documents in a period are aggregated according to their respective topic proportion. For example, for a given topic, the topical sentiment in period $t$ is computed using:

$$s_t = \frac{\sum_{d=1}^{D} s_d \times \theta_d}{\sum_{d=1}^{D} \theta_d}$$

where $s_d$ is the sentiment of the topic in document $d$ and $\theta_d$ the topic proportion in a document $d$.

**Value**

an xts::xts or data.frame containing the time series of topical sentiments.

**See Also**

sentopics_sentiment, sentopics_date

Other series functions: proportion_topics(), sentiment_breakdown(), sentiment_series()

**Examples**

```r
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
sentiment_topics(lda)

# plot shortcut
plot_sentiment_topics(lda, period = "month", rolling_window = 3)
# with or without ridgelines
plot_sentiment_topics(lda, period = "month", plot_ridgelines = FALSE)

# also available for rJST models with internal sentiment computation
rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
```
```r
rjst <- fit(rjst, 100)
sentopics_sentiment(rjst, override = TRUE)
sentiment_topics(rjst)
```

## `sentopics_date`

<table>
<thead>
<tr>
<th>sentopics_date</th>
<th>Internal date</th>
</tr>
</thead>
</table>

### Description

Extract or replace the internal dates of a `sentopicmodel`. The internal dates are used to create time series using the functions `sentiment_series()` or `sentiment_topics()`. Dates should be provided by using `sentopics_date(x) <- value` or by storing a `.date` docvars in the `tokens` object used to create the model.

### Usage

```r
sentopics_date(x, include_docvars = FALSE)
sentopics_date(x) <- value
```

### Arguments

- **x**
  - a `sentopicmodel` created from the `LDA()`, `JST()`, `rJST()` or `sentopicmodel()` function
- **include_docvars**
  - if TRUE the function will return all docvars stored in the internal tokens object of the model
- **value**
  - a Date-coercible vector of dates to input into the model.

### Value

a `data.frame` with the stored date per document.

### Note

The internal date is stored internally in the `docvars` of the topic model. This means that dates may also be accessed through the `docvars()` function, although this is discouraged.

### Author(s)

Olivier Delmarcelle

### See Also

Other `sentopics` helpers: `sentopics_labels()`, `sentopics_sentiment()`
Examples

```r
# example dataset already contains ".date" docvar
docvars(ECB_press_conferences_tokens)
# dates are automatically stored in the sentopicmodel object
lda <- LDA(ECB_press_conferences_tokens)
sentopics_date(lda)

# dates can be removed or modified by the assignment operator
sentopics_date(lda) <- NULL
sentopics_date(lda) <- docvars(ECB_press_conferences_tokens, ".date")
```

---

**sentopics_labels**

Setting topic or sentiment labels

**Description**

Extract or replace the labels of a sentopicmodel. The replaced labels will appear in most functions dealing with the output of the sentomicmodel.

**Usage**

```r
sentopics_labels(x, flat = TRUE)
sentopics_labels(x) <- value
```

**Arguments**

- `x`: a sentopicmodel created from the `LDA()`, `JST()`, `rJST()` or `sentopicmodel()` function
- `flat`: if FALSE, return a list of dimension labels instead of a character vector.
- `value`: a list of future labels for the topic model. The list should be named and contain a character vector for each dimension to label. See the examples for a correct usage.

**Value**

a character vector of topic/sentiment labels.

**Author(s)**

Olivier Delmarcelle

**See Also**

`mergeTopics`

Other sentopics helpers: `sentopics_date()`, `sentopics_sentiment()`
Examples

# by default, sentopics_labels() generate standard topic names
lda <- LDA(ECB_press_conferences_tokens)
sentopics_labels(lda)

# to change labels, a named list must be provided
sentopics_labels(lda) <- list(
  topic = paste0("superTopic", 1:lda$K)
)
sentopics_labels(lda)

# using NULL remove labels
sentopics_labels(lda) <- NULL
sentopics_labels(lda)

# also works for JST/rJST models
jst <- JST(ECB_press_conferences_tokens)
sentopics_labels(jst) <- list(
  topic = paste0("superTopic", 1:jst$K),
  sentiment = c("negative", "neutral", "positive")
)
sentopics_labels(jst)

# setting flat = FALSE return a list or labels for each dimension
sentopics_labels(jst, flat = FALSE)

sentopics_sentiment  Internal sentiment

Description

Compute, extract or replace the internal sentiment of a sentopicmodel. The internal sentiment is used to create time series using the functions `sentiment_series()` or `sentiment_topics()`. If the input model contains a sentiment layer, sentiment can be computed directly from the output of the model. Otherwise, sentiment obtained externally should be added for each document.

Usage

sentopics_sentiment(
  x,
  method = c("proportional", "proportionalPol"),
  override = FALSE,
  quiet = FALSE,
  include_docvars = FALSE
)
sentopics_sentiment(x) <- value
sentopic_sentiment

Arguments

- **x**: A `sentopicmodel` created from the `LDA()`, `JST()`, `rJST()` or `sentopicmodel()` function.
- **method**: The method used to compute sentiment, see "Methods" below. Ignored if an internal sentiment is already stored, unless `override` is `TRUE`.
- **override**: By default, the function computes sentiment only if no internal sentiment is already stored within the `sentopicmodel` object. This avoids, for example, erasing externally provided sentiment. Set to `TRUE` to force computation of new sentiment values. Only useful for models with a sentiment layer.
- **quiet**: If `FALSE`, print a message when internal sentiment is found.
- **include_docvars**: If `TRUE`, the function will return all docvars stored in the internal `tokens` object of the model.
- **value**: A numeric vector of sentiment to input into the model.

Details

The computed sentiment varies depending on the model. For `LDA`, sentiment computation is not possible.

For `JST`, the sentiment is computed on a per-document basis according to the document-level sentiment mixtures.

For a `rJST` model, a sentiment is computed for each topic, resulting in $K$ sentiment values per document. In that case, the `.sentiment` column is an average of the $K$ sentiment values, weighted by their respective topical proportions.

Value

A `data.frame` with the stored sentiment per document.

Methods

The function accepts two methods of computing sentiment:

- **proportional**: Computes the difference between the estimated positive and negative proportions for each document (and possibly each topic).

\[
positive - negative
\]

- **proportionalPol**: Computes the difference between positive and negative proportions, divided by the sum of positive and negative proportions. As a result, the computed sentiment lies within the (-1:1) interval.

\[
\frac{positive - negative}{positive + negative}
\]

Both methods will lead to the same result for a JST model containing only negative and positive sentiments.
Note

The internal sentiment is stored internally in the `docvars` of the topic model. This means that sentiment may also be accessed through the `docvars()` function, although this is discouraged.

Author(s)

Olivier Delmarcelle

See Also

Other sentopics helpers: `sentopics_date()`, `sentopics_labels()`

Examples

```r
# example dataset already contains ".sentiment" docvar
docvars(ECB_press_conferences_tokens)
# sentiment is automatically stored in the sentopicmodel object
lda <- LDA(ECB_press_conferences_tokens)
sentopics_sentiment(lda)

# sentiment can be removed or modified by the assignment operator
sentopics_sentiment(lda) <- NULL
sentopics_sentiment(lda) <- docvars(ECB_press_conferences_tokens, ".sentiment")

# for JST models, sentiment can be computed from the output of the model
jst <- JST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
jst <- fit(jst, 100)
sentopics_sentiment(jst, override = TRUE) # replace existing sentiment

## for rJST models one sentiment value is computed by topic
rjst <- rJST(ECB_press_conferences_tokens, lexicon = LoughranMcDonald)
rjst <- fit(rjst, 100)
sentopics_sentiment(rjst, override = TRUE)
```

topWords

Extract the most representative words from topics

Description

Extract the top words in each topic/sentiment from a sentopicmodel.

Usage

```r
topWords(
  x,
  nWords = 10,
  method = c("frequency", "probability", "term-score", "FREX"),
  output = c("data.frame", "plot", "matrix"),
  subset,
```

w = 0.5
)

plot_topWords(
  x,
  nWords = 10,
  method = c("frequency", "probability", "term-score", "FREX"),
  subset,
  w = 0.5
)

Arguments

x         a sentopicmodel created from the LDA(), JST() or rJST()
nWords    the number of top words to extract
method    specify if a re-ranking function should be applied before returning the top words. See Details for a description of each method.
output    determines the output of the function
subset    allows to subset using a logical expression, as in subset(). Particularly useful to limit the number of observation on plot outputs. The logical expression uses topic and sentiment indices rather than their label. It is possible to subset on both topic and sentiment but adding a & operator between two expressions.
w         only used when method = "FREX". Determines the weight assigned to the exclusivity score at the expense of the frequency score.

Details

"frequency" ranks top words according to their frequency within a topic. This method also reports the overall frequency of each word. When returning a plot, the overall frequency is represented with a grey bar.
"probability" uses the estimated topic-word mixture φ to rank top words.
"term-score" implements the re-ranking method from Blei and Lafferty (2009). This method down-weights terms that have high probability in all topics using the following score:

\[
\text{term-score}_{k,v} = \phi_{k,v} \log \left( \frac{\phi_{k,v}}{\prod_{j=1}^{K} \phi_{j,v}} \right),
\]

for topic k, vocabulary word v and number of topics K.
"FREX" implements the re-ranking method from Bischof and Airoldi (2012). This method used the weight w to balance between topic-word probability and topic exclusivity using the following score:

\[
\text{FREX}_{k,v} = \left( \frac{w}{\text{ECDF} \left( \frac{\phi_{k,v}}{\sum_{j=1}^{K} \phi_{j,v}} \right)} + \frac{1 - w}{\text{ECDF} (\phi_{k,v})} \right),
\]

for topic k, vocabulary word v, number of topics K and weight w, where ECDF is the empirical cumulative distribution function.
Value

The top words of the topic model. Depending on the output chosen, can result in either a long-style data.frame, a ggplot2 object or a matrix.

Author(s)

Olivier Delmarcelle

References


See Also

melt.sentopicmodel() for extracting estimated mixtures

Examples

model <- LDA(ECB_press_conferences_tokens)
model <- fit(model, 10)
topWords(model)
topWords(model, output = "matrix")
topWords(model, method = "FREX")
plot_topWords(model)
plot_topWords(model, subset = topic %in% 1:2)

jst <- JST(ECB_press_conferences_tokens)
jst <- fit(jst, 10)
plot_topWords(jst)
plot_topWords(jst, subset = topic %in% 1:2 & sentiment == 3)
Index

* datasets
  ECB_press_conferences, 12
  ECB_press_conferences_tokens, 13
  LoughranMcDonald, 21
  PicaultRenault, 24
  PicaultRenault_data, 25

* sentopics helpers
  sentopics_date, 39
  sentopics_labels, 40
  sentopics_sentiment, 41

* series functions
  proportion_topics, 29
  sentiment_breakdown, 33
  sentiment_series, 35
  sentiment_topics, 37

* topic models
  JST, 17
  LDA, 19
  rJST, 31

  as.LDA, 4
  as.LDA_lda (as.LDA), 4
  as.tokens.dfm, 6

  chainsDistances, 7
  chainsDistances(), 9, 26, 27
  chainsScores, 8
  chainsScores(), 8
  cmdscale, 27
  coherence, 9
  coherence(), 6, 9

  compute_PicaultRenault_scores, 11
  compute_PicaultRenault_scores(), 25

  data.table, 16, 22, 23, 25
  data.table::dcast, 23
  data.table::melt, 22, 23
  dfm, 6, 11
  docvars, 39, 43

  ECB_press_conferences, 12, 13

  ECB_press_conferences_tokens, 12, 13
  fit(), 5, 7, 9, 10, 18–20, 22, 24, 26, 27, 31–33
  fit.JST (fit.sentopicmodel), 14
  fit.LDA (fit.sentopicmodel), 14
  fit.multiChains (fit.sentopicmodel), 14
  fit.rJST (fit.sentopicmodel), 14
  fit.sentopicmodel, 14
  future.apply::future_lapply, 9, 15
  future::plan, 9, 15

  get_ECB_press_conferences, 15
  get_ECB_speeches, 16
  grow (fit.sentopicmodel), 14

  JST, 17, 20, 29, 33, 42
  JST(), 10, 14, 15, 21, 22, 27, 29–31, 36, 39, 40, 42, 44

  LDA, 17–19, 19, 29, 32, 33, 42
  LDA(), 5, 10, 14, 15, 22–24, 27, 29–31, 34, 36, 37, 39, 40, 42, 44

  LDAvis, 20
  LDAvis(), 27
  LDAvis::createJSON, 20
  LDAvis::serVis, 20
  lexicon::hash_valence_shifters, 21
  LoughranMcDonald, 21

  mean(), 30, 34, 36, 38
  melt, 22
  melt.sentopicmodel, 22
  melt.sentopicmodel(), 22, 45
  mergeTopics, 23
  mergeTopics(), 5

  PicaultRenault, 11, 24
  PicaultRenault_data, 25
  plot.multiChains, 26
  plot.multiChains(), 8
  plot.sentopicmodel, 27
plot.sentopicmodel(), 20
plot_proportion_topics
  (proportion_topics), 29
plot_sentiment_breakdown
  (sentiment_breakdown), 33
plot_sentiment_topics
  (sentiment_topics), 37
plot_topWords (topWords), 43
plotly::plotly, 27
print.JST (print.sentopicmodel), 28
print.LDA (print.sentopicmodel), 28
print.rJST (print.sentopicmodel), 28
print.sentopicmodel, 28
proportion_topics, 29, 35, 36, 38

quanteda::as.tokens(), 7
quanteda::corpus, 11, 12
quanteda::corpus_reshape, 11
quanteda::dfm, 6
quanteda::dfm(), 7
quanteda::dfm_trim(), 6
quanteda::dictionary, 21
quanteda::tokens, 6, 13, 16
reset, 30
reset(), 15
rJST, 18, 20, 29, 31, 42
rJST(), 10, 14, 15, 21–24, 27, 29, 31, 34, 36,
  37, 39, 40, 42, 44
rJST.LDA(), 5

sd(), 30, 34, 36, 38
sentiment_breakdown, 30, 33, 36, 38
sentiment_series, 30, 35, 35, 38
sentiment_series(), 33, 39, 41
sentiment_topics, 30, 35, 36, 37
sentiment_topics(), 39, 41
sentometrics::compute_sentiment(), 16
sentopicmodel, 18, 20, 33
sentopicmodel(), 39, 40, 42
sentopics (sentopics-package), 3
sentopics-package, 3
sentopics_date, 39, 40, 43
sentopics_date<- (sentopics_date), 39
sentopics_labels, 39, 40, 43
sentopics_labels<- (sentopics_labels),
  40
sentopics_sentiment, 39, 40, 41

spacyr::spacy_parse(), 16
stats::cmdscale(), 26
subset(), 44
textcat::textcat(), 16
tokens, 6, 22, 39
topWords, 43
topWords(), 18–20, 23, 27, 32, 33
xts::xts, 26, 30, 34–36, 38
zoo::rollapply(), 30, 34, 36, 38