Package: sentopics (via r-universe)

October 30, 2026

Type Package

Title Tools for Joint Sentiment and Topic Analysis of Textual Data

Version 1.1.0

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 (>= 1.15.0), RcppHungarian

Suggests ggplot2, ggridges, plotly, RColorBrewer, xts, zoo, future, future.apply, progressr, progress, testthat, covr, stm, lda, topicmodels, seededlda (>= 1.4.3), keyATM, LDAvis, servr, textcat, stringr, sentometrics, spacyr, knitr, rmarkdown, webshot

LinkingTo Rcpp, RcppArmadillo, RcppProgress

RcppModules model_module

RoxygenNote 7.3.3

LazyData true

VignetteBuilder knitr

NeedsCompilation yes

Author Olivier Delmarcelle [aut, cre] (ORCID: <https://orcid.org/0000-0003-4347-070X>), Samuel Borms [ctb] (ORCID: <https://orcid.org/0000-0001-9533-1870>), Chenghua 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(), ORCID: <https://orcid.org/0000-0002-3671-836X>)

Repository https://cran.r-universe.dev

Date/Publication 2026-10-30 14:27:55 UTC

RemoteUrl https://github.com/cran/sentopics

RemoteRef HEAD

RemoteSha 8812086b5e0e948769b4c2d7951037d1dfef344e

Contents

sentopics-package . . . 3
as.LDA . . . 4
as.tokens.dfm . . . 6
chains_distances . . . 7
chains_scores . . . 9
coherence . . . 10
compute_PicaultRenault_scores . . . 11
ECB_press_conferences . . . 12
ECB_press_conferences_tokens . . . 13
fit.sentopicsmodel . . . 14
get_ECB_press_conferences . . . 16
get_ECB_speeches . . . 17
JST . . . 18
LDA . . . 20
LDAvis . . . 21
LoughranMcDonald . . . 22
melt.sentopicsmodel . . . 23
merge_topics . . . 24
PicaultRenault . . . 25
PicaultRenault_data . . . 26
plot.multi_chains . . . 27
plot.sentopicsmodel . . . 28
print.sentopicsmodel . . . 29
proportion_topics . . . 29

reset 31
rJST 32
sentiment_breakdown 34
sentiment_series 36
sentiment_topics 38
sentopics_date 40
sentopics_labels 41
sentopics_sentiment 42
top_words 45

Index **48**

`sentopics-package` *Tools for joining sentiment and topic analysis (sentopics)*

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]
- Chenghua 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
- Report bugs at 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

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

## S3 method for class 'STM'
as.LDA(x, docs, count_method = c("heuristic", "optimized"), ...)

## S3 method for class 'LDA_Gibbs'
as.LDA(x, docs, ...)

## S3 method for class 'LDA_VEM'
as.LDA(x, docs, count_method = c("heuristic", "optimized"), ...)

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

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

## S3 method for class 'keyATM_output'
as.LDA(x, docs, count_method = c("heuristic", "optimized"), ...)
```

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.
`count_method`	how integer counts are reconstructed for **stm**, variational **topicmodels**, and **keyATM** models. `"heuristic"` (the default) uses the original fast round-then-repair strategy with corrected residual selection and a unit-donor fallback. `"optimized"` minimizes squared error in the reconstructed conditional probabilities at greater computational cost. Both preserve the required document, word and topic totals.

`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()`, `merge_topics()` 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.

With `count_method = "heuristic"`, any remaining count deficits are filled using `stats::r2dtable()`. This follows R's current random state; use `set.seed()` before conversion for reproducible results.

With `count_method = "optimized"`, each reconstructed integer count table minimizes the sum of squared differences between its implied conditional probabilities and the source probabilities, subject to its required margins. The implied probabilities include the conversion's model priors. This uses a deterministic integer minimum-cost flow computation, up to floating-point precision. Both methods retain the source `theta` and `phi` unchanged.

For a model already created by sentopics, `as.LDA()` is a structural conversion: it requires topics in the first layer and a single inner layer. It does not marginalize or refit sentiment layers. In particular, JST models and rJST models with multiple sentiments are rejected. To estimate an LDA model from their documents, use `LDA(x$tokens)` followed by `fit()`. An existing LDA model is returned unchanged.

Value

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

Examples

```
## 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)
# Opt in to the closest feasible conditional probabilities
as.LDA(stm, docs = poliblog5k.docs, count_method = "optimized")

## 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("quanteda")
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$doc_dfm)
out <- keyATM(docs = keyATM_docs, model = "base",
              no_keyword_topics = 5,
              keywords = keyATM_data_bills$keywords)
LDA <- as.LDA(out, docs = keyATM_docs)
```

`as.tokens.dfm` *Convert back a dfm to a tokens object*

Description

Convert back a dfm to a tokens object

Usage

```
## 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 quanteda::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.

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)
```

---

chains_distances *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

```
chains_distances(
  x,
  method = c("euclidean", "hellinger", "cosine", "minMax", "naiveEuclidean",
    "invariantEuclidean"),
  ...
)
```

Arguments

x	a valid multi_chains 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).
- 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 reverse-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

Tang, J., Meng, Z., Nguyen, X., Mei, Q., and Zhang, M. (2014). Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis. In *Proceedings of the 31st International Conference on Machine Learning*, 32, 90–198.

See Also

plot.multi_chains() chains_scores()

Examples

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

---

chains_scores *Compute scores of topic models (chains)*

Description

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

Usage

```
chains_scores(x, window = NULL, nWords = 10)
```

Arguments

x	a valid multi_chains 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

chains_distances() coherence()

Examples

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

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

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

---

coherence *Coherence of estimated topics*

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`.

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. The maximum distance between two tokens to be considered *co-occurring* for the coherence measure. Distance is expressed in token positions. If `NULL`, use the default window for each coherence metric (10 for C_NPMI and 110 for C_V). Providing an integer or `"boolean"` to this argument will override the default. If `"boolean"`, co-occurences will simply take place if the two words are present in the same document. This argument has no effect if 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

Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 399-–408.

`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

```
compute_PicaultRenault_scores(
  x,
  min_ngram = 2,
  return_dfm = FALSE,
  stopwords = c("a", "the", "an", "of", "to")
)
```

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 quanteda::dfm, on for the Monetary Policy and one for the Economic Condition categories. If `FALSE`, returns the sum of all word scores per document.
`stopwords`	words removed before matching lexicon n-grams. Set to `character(0)` to disable removal. Words are lowercased and stemmed using the same procedure as the input text.

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.

Stop words must be removed before matching because the lexicon was built from stop-word-filtered text. Picault and Renault (2017) report removing 32 stop words, but only name five examples in the paper: "a", "the", "an", "of", and "to". These five words form the default here. The complete original list has not been verified; supply it through `stopwords` if available for exact preprocessing replication. A general English stop list can remove meaningful words retained by the lexicon, including negations, and should not be substituted without checking its contents.

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

Picault, M. & Renault, T. (2017). Words are not all created equal: A new measure of ECB communication. *Journal of International Money and Finance*, 79, 136–156. doi:10.1016/j.jimonfin.2017.09.005

See Also

PicaultRenault

Examples

```
# 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 ECB monetary policy statements*

Description

A corpus of 290 ECB monetary policy statements and related policy briefings from June 1998 to October 2026, split into 4942 paragraphs. The document variable `.date` records the date of each communication.

Usage

```
ECB_press_conferences
```

Format

A quanteda::corpus object.

Details

The corpus is drawn from the ECB introductory-statement indexes. Institutional communications unrelated to monetary policy, slide-only entries, podcast links and separate monetary-policy decision releases accompanying a conference statement are excluded. The July 2021 strategy conference and the August 2007 policy briefing are included. Original item and paragraph identifiers are preserved, so a policy statement can be distinguished from another event on the same date. The sample ends with the statement of 29 October 2026. The selection and reasons for exclusion are recorded in the exclusion list in the GitHub repository's data-preparation sources. These source files are not installed with the package. `get_ECB_press_conferences()` retrieves all indexed communications before this selection.

Source

Annual ECB introductory-statement indexes, 1998 to 2026; for example, https://www.ecb.europa.eu/press/press_conference/monetary-policy-statement/2026/html/index_include.en.html.

See Also

ECB_press_conferences_tokens, `get_ECB_press_conferences()`

Examples

```
quanteda::docvars(ECB_press_conferences)
```

ECB_press_conferences_tokens

Tokenized press conferences

Description

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

- Remove paragraphs with ten or fewer lexical tokens
- 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.

Details

Tokenization and part-of-speech tagging use **spacyr** 1.3.0 with **spaCy** 3.8.7 and the English model `en_core_web_sm` 3.8.0.

This preprocessing is intended for topic analysis. None of the negators in LoughranMcDonald remains in the shipped tokens, so its negation handling has no effect on models initialized from this dataset. For an analysis that needs negation, start from ECB_press_conferences or other original text and retain negators and their neighboring sentiment words until the model's lexicon preprocessing has processed them. Removing stop words or selecting parts of speech first can discard this information.

Source

Annual ECB introductory-statement indexes, 1998 to 2026; for example, https://www.ecb.europa.eu/press/press_conference/monetary-policy-statement/2026/html/index_include.en.html.

See Also

ECB_press_conferences

Examples

```
LDA(ECB_press_conferences_tokens)
```

`fit.sentopicsmodel` *Estimate a topic model*

Description

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

Usage

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

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

Arguments

`object`	a model created with the `LDA()`, `JST()` or `rJST()` function.
`iterations`	a finite nonnegative integer giving the number of additional Gibbs iterations. Zero rebuilds the current output without drawing samples.
`nChains`	if set above 1, the model will be fitted multiple times from various starting positions. Latent variables will be re-initialized if `object` has not been fitted before.
`displayProgress`	if `TRUE`, a progress bar will be displayed indicating the progress of the computation. When `nChains` is greater than 1, this requires the package **progressr** and optionally **progress**.
`computeLikelihood`	if set to `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 `sentopicsmodel` of the relevant model class if `nChains` is unspecified or equal to 1. A `multi_chains` object if `nChains` is greater than 1.

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.

Chain metadata

Tokens, non-sentiment document variables, vocabulary and fixed priors are stored once in the shared core. Each chain stores only its own raw and scaled sentiment columns. Accessing a chain assembles a temporary token view with those columns; computing sentiment does not change another chain's values. `as.list(chains)` makes independent copies; `copy = FALSE` retains references to the shared core and each chain's sentiment cache.

See Also

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

Examples

```
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

Retrieves the HTML pages linked from the ECB's annual introductory-statement indexes and extracts their paragraphs before Q&A. Website navigation and correction notices are removed, and host welcome addresses are kept in separate sections when identified in the page.

Usage

```
get_ECB_press_conferences(
  years = 1998:2026,
  language = "en",
  data.table = TRUE
)
```

Arguments

years	the years for which press conferences should be retrieved
language	the language in which press conferences should be retrieved
data.table	if TRUE, returns a data.table::data.table. Otherwise, return a list in which each element corresponds to an indexed HTML page.

Value

A data.table::data.table of extracted paragraphs when data.table = TRUE, or a list with one element per indexed HTML page.

---

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.

JST *Create a Joint Sentiment/Topic model*

Description

This function initialize a Joint Sentiment/Topic model.

Usage

```
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. Optional valence shifters require the corresponding words and their neighboring polarity words to remain in x; see LoughranMcDonald.
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.

The lexicon must preserve the fitted LDA model's token sequence and vocabulary order. Valence shifters can compound words (for example, `not good`) and therefore change that sequence. Such a transition is rejected because the existing topic assignments no longer identify the same words. In that case, initialize `rJST()` directly from the original tokens with the lexicon and fit the new 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 `top_words()` function easily extract the most probables words of each topic/sentiment.

Author(s)

Olivier Delmarcelle

References

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 375–384.

Lin, C., He, Y., Everson, R. and Ruger, S. (2012). Weakly Supervised Joint Sentiment-Topic Detection from Text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6), 1134—1145. doi:10.1109/TKDE.2011.48.

See Also

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

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

Examples

```
# 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)
```

LDA *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.

The lexicon must preserve the fitted LDA model's token sequence and vocabulary order. Valence shifters can compound words (for example, not good) and therefore change that sequence. Such a transition is rejected because the existing topic assignments no longer identify the same words. In that case, initialize rJST() directly from the original tokens with the lexicon and fit the new 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 top_words() function easily extract the most probables words of each topic/sentiment.

Author(s)

Olivier Delmarcelle

References

Blei, D.M., Ng, A.Y. and Jordan, M.I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993–1022.

See Also

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

Other topic models: `JST()`, `rJST()`, `sentopicsmodel()`

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 an 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
`...`	further arguments passed on to `LDAvis::createJSON()` and `LDAvis::serVis()`.

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.sentopicsmodel()`

Examples

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

`LoughranMcDonald` *Loughran-McDonald lexicon*

Description

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

Usage

```
LoughranMcDonald
```

Format

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

Details

Valence shifters operate on the tokens supplied to the model: recognized sequences of a negator and a polarity word (optionally with an amplifier between them) are compounded and assigned the opposite polarity. Shifters already removed during preprocessing cannot affect sentiment. In particular, ECB_press_conferences_tokens contains no lexicon negators; use the original text with suitable preprocessing when negation matters.

Source

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

References

Loughran, T. & McDonald, B. (2011). When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *The Journal of Finance*, 66(1), 35–65.doi:10.1111/j.15406261.2010.01625.x

See Also

`JST()`, `rJST()`

Examples

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

`melt.sentopicsmodel` *Melt for sentopicsmodels*

Description

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

Usage

```
## S3 method for class 'sentopicsmodel'
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

Value

A data.table::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

`top_words()` 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)
```

`merge_topics` *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

```
merge_topics(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.

For rJST, the sentiment probabilities within a merged theme are also recomputed from pooled sentiment counts and pooled `gamma` priors. For each document, theme B and sentiment s, the new probability is `sum(n[k,s] + gamma[k,s]) / sum(n[k,] + gamma[k,])`, where both sums include all topics k in B and the denominator also sums over sentiments. The theme's topic probability instead uses pooled topic counts and `alpha` priors. These two sets of weights generally differ, so merging topics can change document sentiment, even without further sampling. It is an aggregation of assignments, rather than a sum of the original posterior sentiment contributions. To retain those contributions, use `sentiment_breakdown()` on the original model and sum the desired topics.

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

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

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

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

PicaultRenault *Picault-Renault lexicon*

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

Usage

```
PicaultRenault
```

Format

A data.table::data.table object.

Source

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

References

Picault, M. & Renault, T. (2017). Words are not all created equal: A new measure of ECB communication. *Journal of International Money and Finance*, 79, 136–156. doi:10.1016/j.jimonfin.2017.09.005

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

Usage

```
PicaultRenault_data
```

Format

An xts::xts object.

Source

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

References

Picault, M. & Renault, T. (2017). Words are not all created equal: A new measure of ECB communication. *Journal of International Money and Finance*, 79, 136–156. doi:10.1016/j.jimonfin.2017.09.005

Examples

```
head(PicaultRenault_data)
```

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

Description

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

Usage

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

Arguments

`x`	a valid `multi_chains` 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

`chains_distances()` `cmdscale()`

Examples

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

`plot.sentopicsmodel` *Plot a topic model using Plotly*

Description

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

Usage

```
## S3 method for class 'sentopicsmodel'
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

`top_words()` `LDAvis()`

Examples

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

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

print.sentopicsmodel *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

```
## S3 method for class 'sentopicsmodel'
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 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(). na.rm controls rolling summaries and scaling, not missing document-level sentiment: raw document sentiment must be complete.

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 `LDA()`, `JST()` or `rJST()` function and estimated with `fit()`

Value

a `sentopicsmodel` 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

fit()

Examples

```
model <- LDA(ECB_press_conferences_tokens)
model <- fit(model, 10)
reset(model)
```

---

rJST *Create a Reverse Joint Sentiment/Topic model*

Description

This function initialize a Reverse Joint Sentiment/Topic model.

Usage

```
rJST(x, ...)

## 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. Optional valence shifters require the corresponding words and their neighboring polarity words to remain in x; see LoughranMcDonald.
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.

The lexicon must preserve the fitted LDA model's token sequence and vocabulary order. Valence shifters can compound words (for example, not good) and therefore change that sequence. Such a transition is rejected because the existing topic assignments no longer identify the same words. In that case, initialize rJST() directly from the original tokens with the lexicon and fit the new 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 top_words() function easily extract the most probables words of each topic/sentiment.

Author(s)

Olivier Delmarcelle

References

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 375–384.

Lin, C., He, Y., Everson, R. and Ruger, S. (2012). Weakly Supervised Joint Sentiment-Topic Detection from Text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6), 1134—1145. doi:10.1109/TKDE.2011.48.

See Also

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

Other topic models: `JST()`, `LDA()`, `sentopicsmodel()`

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(). na.rm controls rolling summaries and scaling, not missing document-level sentiment: raw document sentiment must be complete.

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.

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)
scores <- compute_PicaultRenault_scores(ECB_press_conferences)
sentopics_sentiment(lda) <- scores[names(lda$tokens), "EC"]
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) # compute sentiment
plot_sentiment_breakdown(rjst)
```

---

sentiment_series *Compute a sentiment time series*

Description

Compute a sentiment time series based on the internal sentiment and dates of a `sentopicsmodel`. 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(). na.rm controls rolling summaries and scaling, not missing document-level sentiment: raw document sentiment must be complete.

Value

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

Side effects

With scale = TRUE, this function deliberately stores .sentiment_scaled and any .s_*_scaled document variables on the input model, using the normalization parameters of the returned series. These document-level values are reused by sentiment_breakdown() and sentiment_topics(). With scale = FALSE, no scaled values are stored; sentopics_sentiment() can still cache raw sentiment when it is first computed. Use data.table::copy() to obtain an independent model when its cached metadata must remain unchanged.

See Also

sentopics_sentiment sentopics_date

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

Examples

```
lda <- LDA(ECB_press_conferences_tokens)
scores <- compute_PicaultRenault_scores(ECB_press_conferences)
sentopics_sentiment(lda) <- scores[names(lda$tokens), "EC"]
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)
rjst <- fit(rjst, 10)
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

```
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,
  ...
)

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.
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(). na.rm controls rolling summaries and scaling, not missing document-level sentiment: raw document sentiment must be complete.

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.

With period = "identity", the function returns topic-weighted sentiment contributions for each document: $s_d \times \theta_d$. These values are returned before aggregation and division by the sum of topic proportions.

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

```
lda <- LDA(ECB_press_conferences_tokens)
lda <- fit(lda, 100)
scores <- compute_PicaultRenault_scores(ECB_press_conferences)
sentopics_sentiment(lda) <- scores[names(lda$tokens), "EC"]
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)
rjst <- fit(rjst, 100)
sentopics_sentiment(rjst)
sentiment_topics(rjst)
```

---

sentopics_date *Internal date*

Description

Extract or replace the internal dates of a sentopicsmodel. 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

```
sentopics_date(x, include_docvars = FALSE)

sentopics_date(x) <- value
```

Arguments

x	a sentopicsmodel created from the LDA(), JST(), rJST() or sentopicsmodel() 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

```
# example dataset already contains ".date" docvar
docvars(ECB_press_conferences_tokens)
# dates are automatically stored in the sentopicsmodel 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 `sentopicsmodel`. The replaced labels will appear in most functions dealing with the output of the `sentomicmodel`.

Usage

```
sentopics_labels(x, flat = TRUE)

sentopics_labels(x) <- value
```

Arguments

`x`	a `sentopicsmodel` created from the `LDA()`, `JST()`, `rJST()` or `sentopicsmodel()` 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

merge_topics

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 `sentopicsmodel`. 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
```

Arguments

x	a sentopicsmodel created from the LDA(), JST(), rJST() or sentopicsmodel() 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 sentopicsmodel object. This avoid, 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, sentiment is computed from the sentiment probabilities conditional on each topic, resulting in K sentiment values per document in the .s_ columns. The .sentiment column is an average of these values, weighted once by their respective topic 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.

Side effects

Computing model-derived sentiment caches `.sentiment` and, for rJST, `.s_*` document variables on `x` without requiring assignment of the return value. Calls that reuse the cache do not recompute it. The cache belongs to that model or chain. `sentiment_series()` may also cache scaled values.

Raw document sentiment must be complete, including when it was added through `docvars()`. Missing raw values are rejected when read as well as by the replacement function. Use the replacement function to validate external sentiment; `override = TRUE` can recompute sentiment for a fitted JST or rJST model.

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.

When sentiment is computed, the result is cached by modifying the model's `tokens` docvars. If a `.sentiment` docvar already exists and `override = FALSE`, the value is returned without recomputing or modifying the model.

Continuing estimation with positive iterations or resetting a model clears model-derived raw and scaled sentiment, so the next extraction uses the new sampler state. Externally assigned sentiment is retained. A zero-iteration fit preserves an existing sentiment cache.

Author(s)

Olivier Delmarcelle

See Also

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

Examples

```
# sentiment from an external source can be added to the model
lda <- LDA(ECB_press_conferences_tokens)
scores <- compute_PicaultRenault_scores(ECB_press_conferences)
sentopics_sentiment(lda) <- scores[names(lda$tokens), "EC"]
```

```
# using the helper function extracts the stored sentiment
sentopics_sentiment(lda)

# internally, the sentiment is stored in the docvars of the tokens
docvars(lda$tokens, ".sentiment")

# sentiment can be removed or modified by the assignment operator
sentopics_sentiment(lda) <- NULL
sentopics_sentiment(lda) <- scores[names(lda$tokens), "EC"]

# 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) # compute 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)
```

`top_words` *Extract the most representative words from topics*

Description

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

Usage

```
top_words(
  x,
  nWords = 10,
  method = c("frequency", "probability", "term-score", "FREX"),
  output = c("data.frame", "plot", "matrix"),
  subset,
  w = 0.5
)

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

Arguments

x	a sentopicsmodel 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:

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

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:

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

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

Blei, DM. and Lafferty, JD. (2009). Topic models.. In *Text Mining*, chapter 4, 101–124.

Bischof JM. and Airoldi, EM. (2012). Summarizing Topical Content with Word Frequency and Exclusivity.. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, 9–16.

See Also

`melt.sentopicsmodel()` for extracting estimated mixtures

Examples

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

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

Index

∗ **datasets**
ECB_press_conferences, 12
ECB_press_conferences_tokens, 13
LoughranMcDonald, 22
PicaultRenault, 25
PicaultRenault_data, 26
∗ **sentopics helpers**
sentopics_date, 40
sentopics_labels, 41
sentopics_sentiment, 42
∗ **series functions**
proportion_topics, 29
sentiment_breakdown, 34
sentiment_series, 36
sentiment_topics, 38
∗ **topic models**
JST, 18
LDA, 20
rJST, 32

as.LDA, 4
as.LDA.keyATM_output *(*as.LDA*)*, 4
as.LDA.LDA_Gibbs *(*as.LDA*)*, 4
as.LDA.LDA_VEM *(*as.LDA*)*, 4
as.LDA.STM *(*as.LDA*)*, 4
as.LDA.textmodel_lda *(*as.LDA*)*, 4
as.LDA_lda *(*as.LDA*)*, 4
as.tokens.dfm, 6

chains_distances, 7
chains_distances(), *9*, *27*
chains_scores, 9
chains_scores(), *9*
cmdscale(), *27*
coherence, 10
coherence(), *7*, *9*
compute_PicaultRenault_scores, 11
compute_PicaultRenault_scores(), *26*

data.table::copy(), *37*
data.table::data.table, *17*, *23*, *25*
data.table::dcast(), *23*
data.table::melt(), *23*
docvars(), *41*, *44*

ECB_press_conferences, 12, *13*, *14*
ECB_press_conferences_tokens, 13, *13*, *22*

fit(), *5*, *8–10*, *19–21*, *23*, *24*, *27*, *28*, *31–34*
fit.JST *(*fit.sentopicsmodel*)*, 14
fit.LDA *(*fit.sentopicsmodel*)*, 14
fit.multi_chains *(*fit.sentopicsmodel*)*, 14
fit.rJST *(*fit.sentopicsmodel*)*, 14
fit.sentopicsmodel, 14
future.apply::future_lapply, *9*, *16*
future::plan(), *9*, *16*

get_ECB_press_conferences, 16
get_ECB_press_conferences(), *13*
get_ECB_speeches, 17
grow *(*fit.sentopicsmodel*)*, 14

JST, 18, *21*, *30*, *34*, *43*
JST(), *10*, *14–16*, *22*, *23*, *28*, *30*, *31*, *37*, *40*, *41*, *43*, *46*

LDA, *18*, *19*, 20, *20*, *30*, *33*, *34*, *43*
LDA(), *5*, *10*, *14–16*, *23*, *24*, *28*, *30*, *31*, *35*, *37*, *38*, *40*, *41*, *43*, *46*
LDAvis, 21
LDAvis(), *28*
LDAvis::createJSON(), *21*
LDAvis::serVis(), *21*
lexicon::hash_valence_shifters, *22*
LoughranMcDonald, *14*, *18*, 22, *33*

mean(), *30*, *35*, *37*, *39*
melt.sentopicsmodel, 23
melt.sentopicsmodel(), *47*
merge_topics, 24

merge_topics(), *5*

PicaultRenault, *12*, 25
PicaultRenault_data, 26
plot.multi_chains, 27
plot.multi_chains(), *9*
plot.sentopicsmodel, 28
plot.sentopicsmodel(), *21*
plot_proportion_topics *(*proportion_topics*)*, 29
plot_sentiment_breakdown *(*sentiment_breakdown*)*, 34
plot_sentiment_topics *(*sentiment_topics*)*, 38
plot_top_words *(*top_words*)*, 45
plotly::plotly, *28*
print.JST *(*print.sentopicsmodel*)*, 29
print.LDA *(*print.sentopicsmodel*)*, 29
print.rJST *(*print.sentopicsmodel*)*, 29
print.sentopicsmodel, 29
proportion_topics, 29, *36, 37*, *40*

quanteda::as.tokens(), *7*
quanteda::corpus, *11–13*
quanteda::corpus_reshape, *12*
quanteda::dfm, *7*, *11*
quanteda::dfm(), *7*
quanteda::dfm_trim(), *7*
quanteda::dictionary, *22*
quanteda::tokens, *7*, *14*, *17*

reset, 31
reset(), *16*
rJST, *19*, *21*, *30*, 32, *43*
rJST(), *10*, *14–16*, *22–24*, *28*, *30, 31*, *35*, *37, 38*, *40, 41*, *43*, *46*
rJST.LDA(), *5*

sd(), *30*, *35*, *37*, *39*
sentiment_breakdown, *31*, 34, *37*, *40*
sentiment_breakdown(), *24*, *37*
sentiment_series, *31*, 36, *36*, *40*
sentiment_series(), *34*, *40*, *42*, *44*
sentiment_topics, *31*, *36, 37*, 38
sentiment_topics(), *37*, *40*, *42*
sentometrics::compute_sentiment(), *17*
sentopics *(*sentopics-package*)*, 3
sentopics-package, 3
sentopics_date, 40, *42*, *44*
sentopics_date<- *(*sentopics_date*)*, 40
sentopics_labels, 41, *41*, *44*
sentopics_labels<- *(*sentopics_labels*)*, 41
sentopics_sentiment, *41*, 42, *42*
sentopics_sentiment(), *37*
sentopics_sentiment<- *(*sentopics_sentiment*)*, 42
sentopicsmodel, *19*, *21*, *34*
sentopicsmodel(), *40, 41*, *43*
set.seed(), *5*
spacyr::spacy_parse(), *17*
stats::cmdscale(), *27*
stats::r2dtable(), *5*
subset(), *46*

textcat::textcat(), *17*
tokens, *7*, *23*, *40*
top_words, 45
top_words(), *19–21*, *23*, *28*, *33, 34*

xts::xts, *26*, *30, 31*, *35–37*, *39*

zoo::rollapply(), *30*, *35*, *37*, *39*