Package 'sentometrics'

May 29, 2018

Type Package

Title An Integrated Framework for Textual Sentiment Time Series Aggregation and Prediction

Version 0.4

Author David Ardia [aut],
Keven Bluteau [aut],
Samuel Borms [aut, cre],
Kris Boudt [aut]

Maintainer Samuel Borms <samuel.borms@unine.ch>

Description Optimized prediction based on textual sentiment, accounting for the intrinsic challenge that sentiment can be computed and pooled across texts and time in various ways. See Ardia et al. (2017) <https://ssrn.com/abstract=3067734>.

Depends R (>= 3.3.0), data.table

License GPL (>= 2)

BugReports https://github.com/sborms/sentometrics/issues

URL https://github.com/sborms/sentometrics

Encoding UTF-8

LazyData true

Suggests testthat, e1071, randomForest

Imports utils, stats, quanteda, stringi, zoo, abind, glmnet, caret, compiler, Rcpp (>= 0.12.13), RcppRoll, ggthemes, ISOweek, MCS, ggplot2, parallel, foreach, doParallel

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-28 22:58:52 UTC

R topics documented:

sentometrics-package *sentometrics: An Integrated Framework for Textual Sentiment Time Series Aggregation and Prediction*

Description

The **sentometrics** package is designed to do time series analysis based on textual sentiment. It accounts for the intrinsic challenge that, for a given text, sentiment can be computed in many ways, as well as the large number of possibilities to pool sentiment across text and time. This additional layer of manipulation does not exist in standard time series analysis and text mining packages. The package also provides an automated means to econometrically model the impact of sentiment in texts on a given variable, by first computing a wide range of textual sentiment time series and then selecting those that are most informative. Altogether, **sentometrics** integrates the *qualification* of sentiment from texts, the *aggregation* into different sentiment measures and the optimized *prediction* based on these measures.

Main functions

- Feature generation: `sento_corpus`, `add_features`
- Sentiment computation and aggregation into sentiment measures: `ctr_agg`, `compute_sentiment`, `sento_measures`, `measures_merge`, `to_global`
- Sparse modelling: `ctr_model`, `sento_model`
- Prediction and post-modelling analysis: `predict.sentomodel`, `retrieve_attributions`, `plot_attributions`, `perform_MCS`

Update

The development version of the package resides at https://github.com/sborms/sentometrics.

Note

The methodology behind the sentiment aggregation framework can be consulted in the working paper "Questioning the news about economic growth: Sparse forecasting using thousands of news-based sentiment values" (Ardia, Bluteau, and Boudt, 2017) at https://ssrn.com/abstract=2976084. The vignette "The R package sentometrics to compute, aggregate and predict with textual sentiment" (Ardia, Bluteau, Borms and Boudt, 2017) at https://ssrn.com/abstract=3067734 provides a hands-on introduction to the methodology and the package's functionalities.

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

Author(s)

Maintainer: Samuel Borms <samuel.borms@unine.ch>

Authors:

- David Ardia <david.ardia@unine.ch>
- Keven Bluteau <keven.bluteau@unine.ch>
- Kris Boudt <kris.boudt@vub.be>

See Also

Useful links:

- https://github.com/sborms/sentometrics
- Report bugs at https://github.com/sborms/sentometrics/issues

---

`add_features` *Add feature columns to a (sento)corpus object*

Description

Adds new feature columns, either user-supplied or based on keyword(s)/regex pattern search, to a provided `sentocorpus` or **quanteda** `corpus` object.

Usage

```
add_features(corpus, featuresdf = NULL, keywords = NULL, do.binary = TRUE,
  do.regex = FALSE)
```

Arguments

`corpus`	a `sentocorpus` object created with `sento_corpus`, or a **quanteda** `corpus` object.
`featuresdf`	a named `data.frame` of type `numeric` where each columns is a new feature to be added to the inputted `corpus` object. If the number of rows in `featuresdf` is not equal to the number of documents in `corpus`, recycling will occur. The numeric values should be between 0 and 1 (included).
`keywords`	a named `list`. For every element, a new feature column is added with a value of 1 for the texts in which (at least one of) the keyword(s) appear(s), and 0 if not (for `do.binary = TRUE`), or with as value the normalized number of times the keyword(s) occur(s) in the text (for `do.binary = FALSE`). If no texts match a keyword, no column is added. The `list` names are used as the names of the new features. For more complex searching, instead of keywords, one can also directly use a single regex expression to define a new feature (cf. the details section).
`do.binary`	a `logical`, cf. argument `keywords`. If `do.binary = FALSE`, the counts are normalized between 0 and 1,
`do.regex`	a `logical` vector equal in length to the number of elements in the `keywords` argument `list`, or a single value if it applies to all. It should be set to `TRUE` at those positions where a single regex expression is used to identify the particular feature.

Details

If a provided feature name is already part of the corpus, it will be replaced. The `featuresdf` and `keywords` arguments can be provided at the same time, or only one of them, leaving the other at `NULL`. The `do.regex` argument points to the corresponding elements in `keywords`. For `FALSE`, we transform the keywords into a simple regex expression, involving `"\b"` for exact word boundary matching and (if multiple keywords) `|` as OR operator. The elements associated to `TRUE` do not undergo the transformation, and are evaluated as given, if the corresponding keywords vector consists of only one expression. Scaling between 0 and 1 is performed via the min-max normalization, per column. Binary features can be used as a mechanism to select the texts which have to be integrated in the respective feature-based sentiment measure(s), but the within-document aggregation still considers the entire corpus in case of `"tf-idf"`, and the option `do.ignoreZeros` should be set to `TRUE` (see `ctr_agg`). Because of this (implicit) selection that can be performed, having complementary features (e.g., `"economy"` and `"noneconomy"`) makes sense.

Value

An updated `corpus` object.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")

# construct a corpus and add (a) feature(s) to it
corpus <- sento_corpus(corpusdf = usnews)
corpus1 <- add_features(corpus,
                        featuresdf = data.frame(random = runif(quanteda::ndoc(corpus))))
corpus2 <- add_features(corpus,
                        keywords = list(pres = "president", war = "war"))
corpus3 <- add_features(corpus,
                        keywords = list(pres = c("Obama", "US president")),
                        do.binary = FALSE)
corpus4 <- add_features(corpus,
                        featuresdf = data.frame(all = 1),
                        keywords = list(pres1 = c("Obama|US [p|P]resident"),
                                        pres2 = c("\\bObama\\b|\\bUS president\\b"),
                                        war = c("war")),
                        do.regex = c(TRUE, TRUE, FALSE))

sum(corpus3$documents$pres) == sum(corpus4$documents$pres2) # TRUE

# adding a complementary feature
nonpres <- data.frame(nonpres = as.numeric(!quanteda::docvars(corpus2)[["pres"]]))
corpus2 <- add_features(corpus2,
                        featuresdf = nonpres)
```

almons *Compute Almon polynomials*

Description

Computes Almon polynomial weighting curves. Handy to self-select specific time aggregation weighting schemes for input in ctr_agg.

Usage

```
almons(n, orders = 1:3, do.inverse = TRUE, do.normalize = TRUE)
```

Arguments

n	a single numeric to indicate the length of the curve (the number of lags, cf., *n*).
orders	a numeric vector as the sequence of the Almon orders (cf., *b*). The maximum value corresponds to *B*.
do.inverse	TRUE if the inverse Almon polynomials should be calculated as well.
do.normalize	TRUE if polynomials should be normalized to unity.

Details

The Almon polynomial formula implemented is: $(1 - (i/n)^b)(i/n)^{B-b}$, where i is the lag index ordered from n to 1. The inverse is computed by changing i/n to $1 - i/n$.

Value

A data.frame of all Almon polynomial weighting curves, of size length(orders) (times two if do.inverse = TRUE).

See Also

ctr_agg

compute_sentiment *Compute document-level sentiment across features and lexicons*

Description

Given a corpus of texts, computes sentiment per document using the bag-of-words approach based on the lexicons provided and a choice of aggregation across words per document. Relies partly on the **quanteda** package. The scores computed are net sentiment (sum of positive minus sum of negative scores).

Usage

```
compute_sentiment(x, lexicons, how = "proportional", nCore = 1,
  dfm = NULL)
```

Arguments

x	either a sentocorpus object created with sento_corpus, a **quanteda** corpus object, or a character vector. The latter two do not incorporate a date dimension. In case of a corpus object, the numeric columns from the docvars are considered as features over which sentiment will be computed. In case of a character vector, sentiment is only computed across lexicons.
lexicons	output from a setup_lexicons call.
how	a single character vector defining how aggregation within documents should be performed. For currently available options on how aggregation can occur, see get_hows()$words.
nCore	a single numeric at least equal to 1 to indicate the number of cores to use for a parallel sentiment computation. We use the %dopar% construct from the **foreach** package. By default, nCore = 1, which implies no parallelization.
dfm	(optional) an output from a **quanteda** dfm call, such that users can specify their own tokenisation scheme (via tokens) as well as other parameters related to the construction of a document-feature matrix (dfm). Make sure the document-feature matrix is constructed from the texts in the sentocorpus object, otherwise, results will be spurious or errors may occur. Note that valence shifters will not be integrated into the features of a user-provided dfm.

Details

For a separate calculation of positive (resp. negative) sentiment, one has to provide distinct positive (resp. negative) lexicons. This can be done using the do.split option in the setup_lexicons function, which splits out the lexicons into a positive and a negative polarity counterpart. NAs are converted to 0, under the assumption that this is equivalent to no sentiment. By default, if the dfm argument is left unspecified, a document-feature matrix (dfm) is created based on a tokenisation that removes punctuation, numbers, symbols and separators, but does not remove stopwords. The number of words for each document is computed based on that same tokenisation. All tokens are converted to lowercase, in line with what the setup_lexicons function does for the lexicons and valence shifters.

Value

A list containing:

corpus	the supplied x object, transformed into a corpus if a character vector.
sentiment	the sentiment scores data.table with a "word_count" column and all lexicon–feature sentiment scores columns.
features	a character vector of the different features.
lexicons	a character vector of the different lexicons used.
howWithin	the supplied how argument.

The last three elements are only present if x is a sentocorpus object. In that case, the "sentiment" data.table also has a "date" column, meaning it can be used for further aggregation into sentiment time series with the perform_agg function.

Author(s)

Samuel Borms

See Also

dfm, tokens

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

l1 <- list_lexicons[c("LM_en", "HENRY_en")]
l2 <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])

# from a sentocorpus object
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 200)
sent <- compute_sentiment(corpusSample, l1, how = "counts")

# from a character vector
sent <- compute_sentiment(usnews[["texts"]][1:200], l1, how = "counts")

## Not run:
# from a corpus object, parallelized
corpusQ <- quanteda::corpus(usnews, text_field = "texts")
sent <- compute_sentiment(corpusQ, l2, how = "counts", nCore = 2)
## End(Not run)

## Not run:
# using a user-supplied dfm with default settings
tok <- quanteda::tokens_tolower(quanteda::tokens(corpus))
dfm <- quanteda::dfm(tok, verbose = FALSE)
sent <- compute_sentiment(corpus, l1, how = "counts", dfm = dfm)
## End(Not run)
```

---

ctr_agg *Set up control for aggregation into sentiment measures*

Description

Sets up control object for aggregation of document-level textual sentiment into textual sentiment measures (indices).

Usage

```
ctr_agg(howWithin = "proportional", howDocs = "equal_weight",
  howTime = "equal_weight", do.ignoreZeros = TRUE, by = "day", lag = 1L,
  fill = "zero", alphasExp = seq(0.1, 0.5, by = 0.1), ordersAlm = 1:3,
  do.inverseAlm = TRUE, weights = NULL, nCore = 1, dfm = NULL, ...)
```

Arguments

howWithin	a single character vector defining how aggregation within documents will be performed. Should length(howWithin) > 1, the first element is used. For currently available options on how aggregation can occur, see get_hows()$words.
howDocs	a single character vector defining how aggregation across documents per date will be performed. Should length(howDocs) > 1, the first element is used. For currently available options on how aggregation can occur, see get_hows()$docs.
howTime	a character vector defining how aggregation across dates will be performed. More than one choice is possible. For currently available options on how aggregation can occur, see get_hows()$time.
do.ignoreZeros	a logical indicating whether zero sentiment values have to be ignored in the determination of the document weights while aggregating across documents. By default do.ignoreZeros = TRUE, such that documents with a raw sentiment score of zero or for which a given feature indicator is equal to zero are considered irrelevant.
by	a single character vector, either "day", "week", "month" or "year", to indicate at what level the dates should be aggregated. Dates are displayed as the first day of the period, if applicable (e.g., "2017-03-01" for March 2017).
lag	a single integer vector, being the time lag to be specified for aggregation across time. By default equal to 1L, meaning no aggregation across time.
fill	a single character vector, one of c("zero", "latest", "none"), to control how missing sentiment values across the continuum of dates considered are added. This impacts the aggregation across time, applying the measures_fill function before aggregating, except if fill = "none". By default equal to "zero", which sets the scores (and thus also the weights) of the added dates to zero in the time aggregation.
alphasExp	a numeric vector of all exponential smoothing factors to calculate weights for, used if "exponential" %in% howTime. Values should be between 0 and 1 (both excluded).
ordersAlm	a numeric vector of all Almon polynomial orders to calculate weights for, used if "almon" %in% howTime.
do.inverseAlm	a logical indicating if for every Almon polynomial its inverse has to be added, used if "almon" %in% howTime.
weights	an optional own weighting scheme, always used if provided as a data.frame with the number of rows equal to the desired lag. The automatic Almon polynomials are created sequentially; if the user wants only specific of such time weighting series it can use almons, select the columns it requires, combine it into a data.frame and supply it under this argument (see examples).

nCore	a single numeric at least equal to 1 to indicate the number of cores to use for a parallel sentiment computation. We use the %dopar% construct from the **foreach** package. By default, nCore = 1, which implies no parallelization.
dfm	(optional) see compute_sentiment.
...	not used.

Details

For currently available options on how aggregation can occur (via the howWithin, howDocs and howTime arguments), call get_hows.

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

measures_fill, almons, compute_sentiment

Examples

```
# simple control function
ctr1 <- ctr_agg(howTime = "linear", by = "year", lag = 3)

# more elaborate control function (particular attention to time weighting schemes)
ctr2 <- ctr_agg(howWithin = "proportionalPol",
                howDocs = "proportional",
                howTime = c("equal_weight", "linear", "almon", "exponential", "own"),
                do.ignoreZeros = TRUE,
                by = "day",
                lag = 20,
                ordersAlm = 1:3,
                do.inverseAlm = TRUE,
                alphasExp = c(0.20, 0.50, 0.70, 0.95),
                weights = data.frame(myWeights = runif(20)))

# set up control function with one linear and two chosen Almon weighting schemes
a <- almons(n = 70, orders = 1:3, do.inverse = TRUE, do.normalize = TRUE)
ctr3 <- ctr_agg(howTime = c("linear", "own"), by = "year", lag = 70,
                weights = data.frame(a1 = a[, 1], a2 = a[, 3]))
```

ctr_merge *Set up control for merging sentiment measures*

Description

Sets up control object for the optional merging (additional aggregation) of sentiment measures as done by measures_merge.

Usage

```
ctr_merge(sentomeasures, features = NA, lexicons = NA, time = NA,
  do.keep = FALSE)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures. This is necessary to check whether the other input arguments make sense.
features	a list with unique features to merge at given name, e.g., list(feat12 = c("feat1", "feat2")). See sentomeasures$features for the exact names to use. Use NA to apply no merging across this dimension.
lexicons	a list with unique lexicons to merge at given name, e.g., list(lex12 = c("lex1", "lex2")). See sentomeasures$lexicons for the exact names to use. Use NA to apply no merging across this dimension.
time	a list with unique time weighting schemes to merge at given name, e.g., list(tw12 = c("tw1", "tw2")). See sentomeasures$time for the exact names to use. Use NA to apply no merging across this dimension.
do.keep	a logical indicating if the original sentiment measures should be kept (i.e., the merged sentiment measures will be added to the current sentiment measures as additional indices if do.keep = TRUE).

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms

See Also

measures_merge

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# set up a correct control function
ctrMerge <- ctr_merge(sentomeasures,
                      time = list(W = c("equal_weight", "linear")),
                      lexicons = list(LEX = c("LM_en", "HENRY_en")),
                      features = list(journals = c("wsj", "wapo")),
                      do.keep = TRUE)

## Not run:
# produces an informative error message
ctrMerge <- ctr_merge(sentomeasures,
                      time = list(W = c("equal_weight", "almon1")),
                      lexicons = list(LEX = c("LM_en", "HENRY_en")),
                      features = list(journals = c("notInHere", "wapo")))
## End(Not run)
```

`ctr_model` *Set up control for sentiment measures-based regression modelling*

Description

Sets up control object for linear or nonlinear modelling of a response variable onto a large panel of textual sentiment measures (and potentially other variables). See `sento_model` for details on the estimation and calibration procedure.

Usage

```
ctr_model(model = c("gaussian", "binomial", "multinomial"), type = c("BIC",
  "AIC", "Cp", "cv"), do.intercept = TRUE, do.iter = FALSE, h = 0,
  oos = 0, do.difference = FALSE, alphas = seq(0, 1, by = 0.2),
  nSample = NULL, trainWindow = NULL, testWindow = NULL, start = 1,
  do.progress = TRUE, nCore = 1)
```

Arguments

`model`	a `character` vector with one of the following: `"gaussian"` (linear regression), `"binomial"` (binomial logistic regression), or `"multinomial"` (multinomial logistic regression).

type — a character vector indicating which model calibration approach to use. Supports "BIC", "AIC" and "Cp" (Mallows's Cp) as sparse regression adapted information criteria (cf., "On the 'degrees of freedom' of the LASSO"; Zou, Hastie, Tibshirani et al., 2007), and "cv" (cross-validation based on the train function from the **caret** package). The adapted information criteria are currently only available for a linear regression.

do.intercept — a logical, TRUE by default fits an intercept.

do.iter — a logical, TRUE induces an iterative estimation of models at the given nSample size and performs the associated one-step ahead out-of-sample prediction exercise through time.

h — an integer value that shifts the time series to have the desired prediction setup; h = 0 means no change to the input data (nowcasting assuming data is aligned properly), h > 0 shifts the dependent variable by h periods (i.e. rows) further in time (forecasting), h < 0 shifts the independent variables by h periods.

oos — a non-negative integer to indicate the number of periods to skip from the end of the training sample up to the out-of-sample prediction(s). This is either used in the cross-validation based calibration approach (if type = "cv"), or for the iterative out-of-sample prediction analysis (if do.iter = TRUE). For instance, given t, the (first) out-of-simple prediction is computed at $t + oos + 1$.

do.difference — a logical, TRUE will difference the target variable y supplied in the sento_model function with as lag the absolute value of the h argument, in which case abs(h) > 0 is required. For example, if h = 2, and assuming the y variable is properly aligned date-wise with the explanatory variables denoted by X (the sentiment measures and other in x), the regression will be of $y_(t+2) - y_t$ on X_t. If h = -2, the regression fitted is $y_(t + 2) - y_t$ on X_{t+2}. The argument is always kept at FALSE if the model argument is one of c("binomial", "multinomial").

alphas — a numeric vector of the different alphas to test for during calibration, between 0 and 1. A value of 0 pertains to Ridge regression, a value of 1 to LASSO regression; values in between are pure elastic net. The lambda values tested for are chosen by the glmnet function or set to 10^seq(2, -2, length.out = 100) in case of cross-validation.

nSample — a positive integer as the size of the sample for model estimation at every iteration (ignored if do.iter = FALSE).

trainWindow — a positive integer as the size of the training sample for cross-validation (ignored if type != "cv").

testWindow — a positive integer as the size of the test sample for cross-validation (ignored if type != "cv").

start — a positive integer to indicate at which point the iteration has to start (ignored if do.iter = FALSE). For example, given 100 possible iterations, start = 70 leads to model estimations only for the last 31 samples.

do.progress — a logical, if TRUE progress statements are displayed during model calibration.

nCore — a single numeric at least equal to 1 to indicate the number of cores to use for a parallel iterative model estimation do.iter = TRUE. We use the %dopar% construct from the **foreach** package. By default, nCore = 1, which implies no parallelization. No progress statements are displayed whatsoever when nCore > 1.

For cross-validation models, parallelization can also be carried out for a single-shot model (`do.iter = FALSE`), whenever a parallel backend is set up. See the examples in `sento_model`.

Value

A `list` encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

`sento_model`

Examples

```
# information criterion based model control functions
ctrIC1 <- ctr_model(model = "gaussian", type = "BIC", do.iter = FALSE, h = 0,
                    alphas = seq(0, 1, by = 0.10))
ctrIC2 <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE, h = 4, nSample = 100,
                    do.difference = TRUE, oos = 3)

# cross-validation based model control functions
ctrCV1 <- ctr_model(model = "gaussian", type = "cv", do.iter = FALSE, h = 0,
                    trainWindow = 250, testWindow = 4, oos = 0, do.progress = TRUE)
ctrCV2 <- ctr_model(model = "binomial", type = "cv", h = 0, trainWindow = 250,
                    testWindow = 4, oos = 0, do.progress = TRUE)
ctrCV3 <- ctr_model(model = "multinomial", type = "cv", h = 2, trainWindow = 250,
                    testWindow = 4, oos = 2, do.progress = TRUE)
ctrCV4 <- ctr_model(model = "gaussian", type = "cv", do.iter = TRUE, h = 0, trainWindow = 45,
                    testWindow = 4, oos = 0, nSample = 70, do.progress = TRUE)
```

---

data-defunct *Datasets with defunct names*

Description

These are datasets that have been renamed and the old names removed. Please change your code to use the new names.

Details

The dataset `lexicons` is defunct, use `list_lexicons` instead.

The dataset `valence` is defunct, use `list_valence_shifters` instead.

`diff.sentomeasures` *Differencing of sentiment measures*

Description

Differences the sentiment measures from a `sentomeasures` object.

Usage

```
## S3 method for class 'sentomeasures'
diff(x, lag = 1, differences = 1, ...)
```

Arguments

`x`	a `sentomeasures` object created using `sento_measures`.
`lag`	a `numeric`, see documentation for the generic `diff`.
`differences`	a `numeric`, see documentation for the generic `diff`.
`...`	not used.

Value

A modified `sentomeasures` object, with the measures replaced by the differenced measures as well as updated statistics.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# first-order difference sentiment measures with a lag of two
diffed <- diff(sentomeasures, lag = 2, differences = 1)
```

epu *Monthly Economic Policy Uncertainty Index*

Description

Monthly values of a news-based index of U.S. Economic Policy Uncertainty (EPU) between January 1980 and September 2014, including a binomial and a multinomial example series. For more information on its calculation, see this. Following columns are present:

- date. Date as `"yyyy-mm-01"`.
- index. A `numeric` monthly index value.
- above. A `factor` with value `"above"` if the index is greater than the mean of the entire series, else `"below"`.
- aboveMulti. A `factor` with values `"above+"`, `"above"`, `"below"` and `"below-"` if the index is greater than the 75% quantile and the 50% quantile, or smaller than the 50% quantile and the 25% quantile, respectively and in a mutually exclusive sense.

Usage

```
data("epu")
```

Format

A `data.frame` with 417 rows and 4 columns.

Source

Research on Economic Policy Uncertainty

Examples

```
data("epu", package = "sentometrics")
head(epu)
```

exponentials *Compute exponential weighting curves*

Description

Computes exponential weighting curves. Handy to self-select specific time aggregation weighting schemes for input in `ctr_agg`.

Usage

```
exponentials(n, alphas = seq(0.1, 0.5, by = 0.1))
```

Arguments

n	a single numeric to indicate the length of the curve (the number of lags).
alphas	a numeric vector of decay factors.

Value

A data.frame of exponential weighting curves per value of alphas.

See Also

ctr_agg

---

get_hows *Options supported to perform aggregation into sentiment measures*

Description

Call for information purposes only. Used within ctr_agg to check if supplied aggregation hows are supported.

Usage

```
get_hows()
```

Details

See the package's vignette for a thoughtful explanation of the different aggregation options. The howWithin = "proportionalPol" option divides each document's initial sentiment score by the number of polarized words found in the document (counting each word only once would it appear multiple times), instead of the total number of words which the "proportional" option gives.

Value

A list with the supported aggregation hows for arguments howWithin ("words"), howDows ("docs") and howTime ("time"), to be supplied to ctr_agg.

See Also

ctr_agg

`list_lexicons` *Built-in lexicons*

Description

A `list` containing all built-in lexicons as a `data.table` with two columns: a `x` column with the words, and a `y` column with the polarities. The `list` element names incorporate consecutively the name and language (based on the two-letter ISO code convention as in `stopwords`), and `"_tr"` as suffix if the lexicon is translated. The translation was done via Microsoft Translator through Microsoft Word. Only the entries that conform to the original language entry after retranslation, and those that have actually been translated, are kept. The last condition is assumed to be fulfilled when the translation differs from the original entry. All words are in lowercase. The lexicons are in the format required for further sentiment analysis. The built-in lexicons are the following:

- FEEL_en_tr (French Expanded Emotion Lexicon)
- FEEL_fr
- FEEL_nl_tr
- GI_en (General Inquirer, i.e. Harvard IV-4 combined with Laswell)
- GI_fr_tr
- GI_nl_tr
- HENRY_en (Henry)
- HENRY_fr_tr
- HENRY_nl_tr
- LM_en (Loughran and McDonald)
- LM_fr_tr
- LM_nl_tr

Other immediate lexicon options can be found in the **lexicon** package, more specifically the datasets preceded by `hash_sentiment_`.

Usage

```
data("list_lexicons")
```

Format

A `list` with all built-in lexicons, appropriately named as `"NAME_language(_tr)"` .

Source

FEEL lexicon

GI lexicon

HENRY lexicon

LM lexicon

Examples

```
data("list_lexicons", package = "sentometrics")
list_lexicons[c("FEEL_en_tr", "LM_en")]
```

`list_valence_shifters` *Built-in valence word lists*

Description

A `list` containing all built-in valence word lists, a `data.table` with three columns: a `x` column with the words, a `t` column with the type of valence words, and a `y` column with the values associated to each word and type of valence shifter. The `list` element names indicate the language (based on the two-letter ISO code convention as in `stopwords`) of the valence word list. All non-English word lists are translated via Microsoft Translator through Microsoft Word. Only the entries whose translation differs from the original entry are kept. The valence word lists are in the form required for further sentiment analysis. All words are in lowercase. The built-in valence word lists are available in following languages:

- English (`"en"`)
- French (`"fr"`)
- Dutch (`"nl"`)

Usage

```
data("list_valence_shifters")
```

Format

A `list` with all built-in valence word lists, appropriately named.

Details

The `t` column is coded as follows: `1` for negators, `2` for amplifiers/intensifiers, and `3` for deamplifiers/downtoners. Default scores are -1, 2, and 0.5 respectively. See `setup_lexicons` for details about the application of the valence shifters to the lexicons, and along the same vein, to the corpus.

Source

`hash_valence_shifters` (English valence shifters)

Examples

```
data("list_valence_shifters", package = "sentometrics")
list_valence_shifters["en"]
```

`measures_delete` *Delete sentiment measures*

Description

Deletes all sentiment measures which include either all of the given deletion components combined, or those who's name consist of at least one of the deletion components.

Usage

```
measures_delete(sentomeasures, toDelete, do.combine = TRUE)
```

Arguments

`sentomeasures`	a `sentomeasures` object created using `sento_measures`.
`toDelete`	a `character` vector of the lexicon, feature and time weighting scheme names, to indicate which measures need to be deleted. One can also supply a `list` of such `character` vectors, in which case `do.combine = TRUE` is set automatically, such that the separately specified combinations are deleted.
`do.combine`	a `logical` indicating if only measures for which all (`do.combine = TRUE`) or at least one (`do.combine = FALSE`) from `toDelete` should occur in each sentiment measure's name for deletion. If `do.combine = TRUE`, the `toDelete` argument can only consist of one lexicon, one feature, and one time weighting scheme at maximum.

Value

A modified `sentomeasures` object, with the required sentiment measures deleted, including updated information and statistics, but the original sentiment scores `data.table` untouched.

Author(s)

Samuel Borms

See Also

`measures_select`

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
```

```
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# different deletions
del1 <- measures_delete(sentomeasures, c("equal_weight"))
del2 <- measures_delete(sentomeasures, c("equal_weight", "linear"), do.combine = FALSE)
del3 <- measures_delete(sentomeasures, c("linear", "LM_en"))
del4 <- measures_delete(sentomeasures, list(c("linear", "wsj"), c("linear", "economy")))
```

`measures_fill` *Add and fill missing dates*

Description

Adds missing dates between earliest and latest date of a `sentomeasures` object, such that time series is continuous date-wise. Fills in these dates with either 0, the respective latest non-missing value or `NA`.

Usage

```
measures_fill(sentomeasures, fill = "zero")
```

Arguments

`sentomeasures`	a `sentomeasures` object created using `sento_measures`.
`fill`	an element of `c("zero", "latest", NA)`; the first and last assume missing dates represent zero sentiment, the second assumes missing dates represent constant sentiment.

Value

A modified `sentomeasures` object.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "day", lag = 7)
```

```
sentomeasures <- sento_measures(corpusSample, l, ctr)

# fill measures
f1 <- measures_fill(sentomeasures)
f2 <- measures_fill(sentomeasures, fill = "latest")
f3 <- measures_fill(sentomeasures, fill = NA)
```

`measures_merge` *Merge sentiment measures*

Description

Merge (further aggregate) measures by combining across provided lexicons, features, and time weighting schemes dimensions. The combination occurs by taking the mean of the relevant measures.

Usage

```
measures_merge(ctr)
```

Arguments

`ctr`	output from a `ctr_merge` call.

Value

A modified `sentomeasures` object, with only the sentiment measures required, including updated information and statistics, but the original sentiment scores `data.table` untouched.

Author(s)

Samuel Borms

See Also

`ctr_merge`

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
```

```
sentomeasures <- sento_measures(corpusSample, l, ctr)

# set up control function and perform the merging
ctrMerge <- ctr_merge(sentomeasures,
                      time = list(W = c("equal_weight", "linear")),
                      features = list(journals = c("wsj", "wapo")),
                      do.keep = TRUE)
sentomeasuresMerged <- measures_merge(ctrMerge)
```

`measures_select` *Select sentiment measures*

Description

Selects all sentiment measures which include either all of the given selection components combined, or those who's name consist of at least one of the selection components.

Usage

```
measures_select(sentomeasures, toSelect, do.combine = TRUE)
```

Arguments

`sentomeasures`	a `sentomeasures` object created using `sento_measures`.
`toSelect`	a `character` vector of the lexicon, feature and time weighting scheme names, to indicate which measures need to be selected. One can also supply a `list` of such `character` vectors, in which case `do.combine = TRUE` is set automatically, such that the separately specified combinations are selected.
`do.combine`	a `logical` indicating if only measures for which all (`do.combine = TRUE`) or at least one (`do.combine = FALSE`) from `toSelect` should occur in each sentiment measure's name in the selection. If `do.combine = TRUE`, the `toSelect` argument can only consist of one lexicon, one feature, and one time weighting scheme at maximum.

Value

A modified `sentomeasures` object, with only the sentiment measures required, including updated information and statistics, but the original sentiment scores `data.table` untouched.

Author(s)

Samuel Borms

See Also

`measures_delete`

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# different selections
sel1 <- measures_select(sentomeasures, c("equal_weight"))
sel2 <- measures_select(sentomeasures, c("equal_weight", "linear"), do.combine = FALSE)
sel3 <- measures_select(sentomeasures, c("linear", "LM_en"))
sel4 <- measures_select(sentomeasures, list(c("linear", "wsj"), c("linear", "economy")))
```

---

measures_subset *Subset sentiment measures*

Description

Subsets rows of the sentiment measures based on its columns.

Usage

```
measures_subset(sentomeasures, subset)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.
subset	a logical expression indicating the rows to keep.

Value

A modified sentomeasures object, with only the kept rows, including updated information and statistics, but the original sentiment scores data.table untouched.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# different subsets
sub1 <- measures_subset(sentomeasures, HENRY_en--economy--equal_weight >= 0.01)
sub2 <- measures_subset(sentomeasures,
   date %in% seq(as.Date("2000-01-01"), as.Date("2013-12-01"), by = "month"))
```

---

nmeasures *Get number of sentiment measures*

Description

Returns the number of sentiment measures.

Usage

```
nmeasures(sentomeasures)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.

Value

The number of sentiment measures in the input sentomeasures object.

Author(s)

Samuel Borms

nobs *Get number of observations in the sentiment measures*

Description

Returns the number of data points available for the sentiment measures.

Usage

```
nobs(sentomeasures)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.

Value

The number of rows (observations/data points) in sentomeasures[["measures"]].

Author(s)

Samuel Borms

peakdocs *Extract dates and documents related to sentiment peaks*

Description

This function extracts the dates and documents for which aggregated sentiment is most extreme (lowest, highest or both in absolute terms). The extracted dates are unique, even when, for example, all most extreme sentiment values (for different sentiment measures) occur on only one date.

Usage

```
peakdocs(sentomeasures, sentocorpus, n = 10, type = "both",
  do.average = FALSE)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.
sentocorpus	the sentocorpus object created with sento_corpus, used for the construction of the input sentomeasures object.
n	a numeric value to indicate the number of dates associated to sentiment peaks to extract.

type	a character value, either "pos", "neg" or "both", respectively to look for the n dates related to the most positive, most negative or most extreme (in absolute terms) sentiment occurrences.
do.average	a logical to indicate whether peaks should be selected based on the average sentiment value per date.

Value

A list with as elements "dates", "ids" and "documents", corresponding to the n extracted sentiment peak dates and associated document ids and texts.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "month", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# extract the peaks
peaksAbs <- peakdocs(sentomeasures, corpus, n = 5)
peaksPos <- peakdocs(sentomeasures, corpus, n = 5, type = "pos")
peaksNeg <- peakdocs(sentomeasures, corpus, n = 5, type = "neg")
```

---

perform_agg *Aggregate textual sentiment across documents and time*

Description

Condenses document-level textual sentiment scores into a panel of textual sentiment measures by aggregating across documents and time. This function is called within sento_measures, applied on the output of compute_sentiment.

Usage

```
perform_agg(sentiment, ctr)
```

Arguments

sentiment	output from a compute_sentiment call, computed from a sentocorpus object.
ctr	output from a ctr_agg call. The howWithin and nCore arguments are ignored.

Value

A sentomeasures object.

Author(s)

Samuel Borms, Keven Bluteau

See Also

compute_sentiment, ctr_agg, sento_measures

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# computation of sentiment and aggregation into sentiment measures
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
sent <- compute_sentiment(corpusSample, l, how = "counts")
ctr <- ctr_agg(howTime = c("linear"), by = "year", lag = 3)
sentomeasures <- perform_agg(sent, ctr)
```

---

perform_MCS *Apply model confidence set (MCS) procedure to a selection of models*

Description

Calculates the model confidence set (see "The Model Confidence Set"; Hansen, Lunde and Nason, 2011) as implemented in the **MCS** package, for a set of different sentomodeliter objects.

Usage

```
perform_MCS(models, loss = c("DA", "errorSq", "AD", "accuracy"), ...)
```

Arguments

models	a named list of sentomodeliter objects. All models should be of the same family, being either "gaussian", "binomial" or "multinomial", and have performance data of the same dimensions.
loss	a single character vector, either "DA" (directional *in*accuracy), "errorSq" (squared errors), "AD" (absolute errors) or "accuracy" (*in*accurate class predictions). This argument defines on what basis the model confidence set is calculated. The first three options are available for "gaussian" models, the last option applies only to "binomial" and "multinomial" models.
...	other parameters that can be supplied to the MCSprocedure function. If empty, its default values are used.

Value

An object as returned by the MCSprocedure function.

Author(s)

Samuel Borms

See Also

sento_model, MCSprocedure

Examples

```
## Not run:
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
data("epu", package = "sentometrics")

# construct two sentomeasures objects
corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "1997-01-01" & date < "2014-10-01")
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])

ctr1 <- ctr_agg(howWithin = "tf-idf", howDocs = "proportional",
                howTime = c("equal_weight", "linear"), by = "month", lag = 3)
sentMeas1 <- sento_measures(corpus, l, ctr1)

ctr2 <- ctr_agg(howWithin = "counts", howDocs = "equal_weight",
                howTime = c("equal_weight", "linear"), by = "month", lag = 3)
sentMeas2 <- sento_measures(corpus, l, ctr2)

# prepare y and other x variables
y <- epu[epu$date >= sentMeas1$measures$date[1], ]$index
length(y) == nobs(sentMeas1) # TRUE
x <- data.frame(runif(length(y)), rnorm(length(y))) # two other (random) x variables
colnames(x) <- c("x1", "x2")
```

```
# estimate different type of regressions
ctr1 <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                  h = 0, nSample = 120, start = 50)
out1 <- sento_model(sentMeas1, y, x = x, ctr = ctr1)

ctr2 <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                  h = 0, nSample = 120, start = 50)
out2 <- sento_model(sentMeas1, y, x = NULL, ctr = ctr2)

ctr3 <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                  h = 0, nSample = 120, start = 50)
out3 <- sento_model(sentMeas2, y, x = x, ctr = ctr3)

ctr4 <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                  h = 0, nSample = 120, start = 50)
out4 <- sento_model(sentMeas2, y, x = NULL, ctr = ctr4)

mcs <- perform_MCS(models = list(m1 = out1, m2 = out2, m3 = out3, m4 = out4),
                   loss = "errorSq")
## End(Not run)
```

`plot.sentomeasures` *Plot sentiment measures*

Description

Straightforward plotting method that shows all sentiment measures from the provided `sentomeasures` object in one plot, or the average along one of the lexicons, features and time weighting dimensions. We suggest to make use of the `measures_select` function when you desire to plot only a subset of the sentiment measures.

Usage

```
## S3 method for class 'sentomeasures'
plot(x, group = "all", ...)
```

Arguments

`x`	a `sentomeasures` object created using `sento_measures`.
`group`	a value from `c("lexicons", "features", "time", "all")`. The first three choices display the average of all measures from the same group, in a different color. The choice `"all"` displays every single sentiment measure in a separate color, but this may look visually overwhelming very fast, and can be quite slow.
`...`	not used.

Value

Returns a simple ggplot object, which can be added onto (or to alter its default elements) by using the + operator (see examples). By default, a legend is positioned at the top if there are at maximum twelve line graphs plotted.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# plot sentiment measures
plot(sentomeasures, group = "features")

## Not run:
# adjust appearance of plot
p <- plot(sentomeasures)
p <- p +
  ggthemes::theme_base() +
  scale_x_date(name = "month-year") +
  scale_y_continuous(name = "newName")
p
## End(Not run)
```

---

plot.sentomodeliter *Plot iterative predictions versus realized values*

Description

Displays a plot of all predictions made through the iterative model computation as incorporated in the input `sentomodeliter` object, as well as the corresponding true values.

Usage

```
## S3 method for class 'sentomodeliter'
plot(x, ...)
```

Arguments

x	a sentomodeliter object created using sento_model.
...	not used.

Value

Returns a simple ggplot object, which can be added onto (or to alter its default elements) by using the + operator (see examples).

Author(s)

Samuel Borms

Examples

```
## Not run:
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
data("epu", package = "sentometrics")

# construct a sentomeasures object to start with
corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "2007-01-01" & date < "2014-10-01")
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howWithin = "tf-idf", howDocs = "proportional",
               howTime = c("equal_weight", "linear"),
               by = "month", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

# prepare y variable
y <- epu[epu$date >= sentomeasures$measures$date[1], ]$index
length(y) == nobs(sentomeasures) # TRUE

# estimate regression iteratively based on a sample of 60, skipping first 25 iterations
ctr <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                 h = 0, nSample = 60, start = 26)
out <- sento_model(sentomeasures, y, ctr = ctr)
summary(out)

# plotting
p <- plot(out)
p <- p +
  ggthemes::theme_few()
p
## End(Not run)
```

plot_attributions *Plot prediction attributions at specified level*

Description

Shows a plot of the attributions along the dimension provided, stacked per date.

Usage

```
plot_attributions(attributions, group = "features")
```

Arguments

attributions	an output from a retrieve_attributions call.
group	a value from c("lexicons", "features", "time").

Details

See sento_model for an elaborate modelling example including the calculation and plotting of attributions. This function does not handle the plotting of the attribution of individual documents, since there are often a lot of documents involved and they appear only once at one date (even though a document may contribute to predictions at several dates, depending on the number of lags in the time aggregation).

Value

Returns a simple ggplot object, which can be added onto (or to alter its default elements) by using the + operator (see examples). By default, a legend is positioned at the top if the number of components of the dimension (thus, individual line graphs) is at maximum twelve.

Author(s)

Samuel Borms, Keven Bluteau

predict.sentomodel *Make predictions from a sentomodel object*

Description

Prediction method for sentomodel class, with usage along the lines of predict.glmnet, but simplified in terms of allowed parameters.

Usage

```
## S3 method for class 'sentomodel'
predict(object, newx, type, offset = NULL, ...)
```

Arguments

object	a sentomodel object created with sento_model.
newx	a matrix of numeric values with all explanatory variables to be used for the prediction(s), structured row-by-row; see documentation for predict.glmnet. The number of variables should be equal to sentomodel$nVar, being the sum of the number of original sentiment measures and the number of additional explanatory variables. Variables discarded in the regression process are discarded again here, based on sentomodel$discarded.
type	type of prediction required, a value from c("link", "response", "class"), see documentation for predict.glmnet.
offset	not used. Any values here will be ignored.
...	not used.

Value

A prediction output depending on the type argument.

Author(s)

Samuel Borms

See Also

predict.glmnet, sento_model

---

retrieve_attributions *Retrieve top-down model sentiment attributions*

Description

Computes the attributions to predictions for a (given) number of dates at all possible sentiment dimensions, based on the coefficients associated to each sentiment measure, as estimated in the provided model object.

Usage

```
retrieve_attributions(model, sentomeasures, do.normalize = FALSE,
  refDates = NULL, factor = NULL)
```

Arguments

model	a sentomodel or sentomodeliter object created with sento_model.
sentomeasures	the sentomeasures object, as created with sento_measures, used to estimate the model from the first argument.

do.normalize	a logical, TRUE divides each element of every attribution vector at a given date by its L2-norm at that date, normalizing the values between -1 and 1. The document attributions are not normalized.
refDates	the dates (as "yyyy-mm-dd") at which attribution is to be performed. These should be between the latest date available in the input sentomeasures object and the first estimation sample date, i.e. model$dates[1] if model is a sentomodel object. All dates should also be present in sentomeasures$measures$date. If NULL (default), attribution is calculated for all in-sample dates. Ignored if model is a sentomodeliter object, for which attribution is calculated for all out-of-sample prediction dates.
factor	the factor level as a single character vector for which attribution has to be calculated in case of (a) multinomial model(s). Ignored for linear and binomial models.

Details

See sento_model for an elaborate modelling example including the calculation and plotting of attributions. The attribution for logistic models is represented in terms of log odds. For binomial models, it is calculated with respect to the last factor level or factor column.

Value

A list with all dimensions for which aggregation is computed, being "documents", "lexicons", "features" and "time". The last three dimensions are data.tables having a "date" column and the other columns the different components of the dimension, with the attributions as values. Document-level attribution is further decomposed into a data.table per date, with "id", "date" and "attrib" columns.

Author(s)

Samuel Borms, Keven Bluteau

See Also

sento_model

---

scale.sentomeasures *Scaling and centering of sentiment measures*

Description

Scales and centers the sentiment measures from a sentomeasures object, column-per-column. By default, the measures are normalized. NAs are removed first.

Usage

```
## S3 method for class 'sentomeasures'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	a sentomeasures object created using sento_measures.
center	a logical, see documentation for the generic scale.
scale	a logical, see documentation for the generic scale.

Value

A modified sentomeasures object, with the measures replaced by the scaled measures as well as updated statistics.

Author(s)

Samuel Borms

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpusSample, l, ctr)

# scale sentiment measures
scaled <- scale(sentomeasures)
```

---

sentometrics-deprecated

Deprecated functions

Description

Functions deprecated due to changed naming or because functionality is discarded. Deprecated functions are made defunct every 1 major or every 2 minor package updates. See the NEWS file for more information about since when or why functions have been deprecated.

Usage

```
fill_measures(sentomeasures, fill = "zero")

merge_measures(ctr)
```

```
subset_measures(sentomeasures, subset)

select_measures(sentomeasures, toSelect, do.combine = TRUE)

extract_peakdocs(sentomeasures, sentocorpus, n = 10, type = "both",
  do.average = FALSE)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.
fill	an element of c("zero", "latest", NA); the first and last assume missing dates represent zero sentiment, the second assumes missing dates represent constant sentiment.
ctr	output from a ctr_merge call.
subset	a logical expression indicating the rows to keep.
toSelect	a character vector of the lexicon, feature and time weighting scheme names, to indicate which measures need to be selected. One can also supply a list of such character vectors, in which case do.combine = TRUE is set automatically, such that the separately specified combinations are selected.
do.combine	a logical indicating if only measures for which all (do.combine = TRUE) or at least one (do.combine = FALSE) of the selection components should occur in each sentiment measure's name in the selection. If do.combine = TRUE, the toSelect argument can only consist of one lexicon, one feature, and one time weighting scheme at maximum.
sentocorpus	the sentocorpus object created with sento_corpus, used for the construction of the input sentomeasures object.
n	a numeric value to indicate the number of dates associated to sentiment peaks to extract.
type	a character value, either "pos", "neg" or "both", respectively to look for the n dates related to the most positive, most negative or most extreme (in absolute terms) sentiment occurrences.
do.average	a logical to indicate whether peaks should be selected based on the average sentiment value per date.

See Also

measures_fill

measures_merge

measures_subset

measures_select

peakdocs

sento_corpus *Create a sentocorpus object*

Description

Formalizes a collection of texts into a well-defined corpus object, by mainly calling the corpus function from the **quanteda** package. This package provides a fast text mining infrastructure; for more info, see quanteda. Their formal corpus structure is required for better memory management, corpus manipulation, and sentiment calculation. This function mainly performs a set of checks on the input data and prepares the corpus for further sentiment analysis.

Usage

```
sento_corpus(corpusdf, do.clean = FALSE)
```

Arguments

corpusdf	a data.frame (or a data.table, or a tbl) with as named columns: a document "id" column, a "date" column, a "texts" column (i.e., the columns where all texts to analyze reside), and a series of feature columns of type numeric, with values pointing to the applicability of a particular feature to a particular text. The latter columns can be binary (1 means the feature is applicable to the document in the same row) or a value between 0 and 1 to specify the degree of connectedness of a feature to a document. Features could be topics (e.g., legal, political, or economic), but also article sources (e.g., online or printed press), amongst many more options. If you have no knowledge about features or no particular features are of interest to your analysis, provide no feature columns. In that case, the corpus constructor automatically adds an additional feature column named "dummy". Provide the date column as "yyyy-mm-dd". The id column should be in character mode. All spaces in the names of the features are replaced by underscores. If the feature columns have values not between 0 and 1, they will be rescaled column-wise and a warning will be issued.
do.clean	a logical, if TRUE all texts undergo a cleaning routine to eliminate common textual garbage. This includes a brute force replacement of HTML tags and non-alphanumeric characters by an empty string. To use with care if the text is meant to have non-alphanumeric characters! Preferably, cleaning is done outside of this function call.

Details

A sentocorpus object is a specialized instance of a **quanteda** corpus. In theory, all **quanteda** functions applicable to its corpus object can also be applied to a sentocorpus object. However, changing a given sentocorpus object too drastically using some of **quanteda**'s functions might alter the very structure the corpus is meant to have (as defined in the corpusdf argument) to be able to be used as an input in other functions of the **sentometrics** package. There are functions, including corpus_sample or corpus_subset, that do not change the actual corpus structure and may come in handy. To add additional features, use add_features.

Value

A sentocorpus object, derived from a **quanteda** corpus classed list with the elements "documents", "metadata", and "settings" kept. The first element incorporates the corpus represented as a data.frame.

Author(s)

Samuel Borms

See Also

corpus, add_features

Examples

```
data("usnews", package = "sentometrics")

# corpus construction
corpus <- sento_corpus(corpusdf = usnews)

# take a random subset making use of quanteda
corpusSmall <- quanteda::corpus_sample(corpus, size = 500)

# deleting a feature
quanteda::docvars(corpus, field = "wapo") <- NULL

# corpus creation when no features are present
corpusDummy <- sento_corpus(corpusdf = usnews[, 1:3])
```

---

sento_measures *One-way road towards a sentomeasures object*

Description

Wrapper function which assembles calls to compute_sentiment and perform_agg, and includes the input sentocorpus and computed sentiment scores in its output. Serves as the most direct way towards a panel of textual sentiment measures as a sentomeasures object.

Usage

```
sento_measures(sentocorpus, lexicons, ctr)
```

Arguments

sentocorpus	a sentocorpus object created with sento_corpus.
lexicons	output from a setup_lexicons call.
ctr	output from a ctr_agg call.

Value

A sentomeasures object, which is a list containing:

measures	a data.table with a "date" column and all textual sentiment measures as remaining columns.
features	a character vector of the different features.
lexicons	a character vector of the different lexicons used.
time	a character vector of the different time weighting schemes used.
by	a single character vector specifying the time interval of aggregation used.
stats	a data.frame with a series of elementary statistics (mean, standard deviation, maximum, minimum, and average correlation with all other measures) for each individual sentiment measure.
sentiment	the sentiment scores data.table with "date", "word_count" and lexicon–feature sentiment scores columns. If ctr$do.ignoreZeros = TRUE, all zeros are replaced by NA.
howWithin	a single character vector to remind how sentiment within documents was aggregated.
howDocs	a single character vector to remind how sentiment across documents was aggregated.
fill	a single character vector that specifies if and how missing dates have been added before aggregation across time was carried out.
do.ignoreZeros	a single character vector to remind if documents with a zero feature-sentiment score have been ignored in the within-document aggregation.
attribWeights	a list of document and time weights used in the retrieve_attributions function. Serves further no direct purpose.

Author(s)

Samuel Borms, Keven Bluteau

See Also

compute_sentiment, perform_agg

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howWithin = "tf-idf",
               howDocs = "proportional",
```

```
                howTime = c("equal_weight", "linear", "almon"),
                by = "month",
                lag = 3,
                ordersAlm = 1:3,
                do.inverseAlm = TRUE)
sentomeasures <- sento_measures(corpusSample, l, ctr)
summary(sentomeasures)
```

`sento_model` *Optimized and automated sparse regression*

Description

Linear or nonlinear penalized regression of any dependent variable on the wide number of sentiment measures and potentially other explanatory variables. Either performs a regression given the provided variables at once, or computes regressions sequentially for a given sample size over a longer time horizon, with associated one-step ahead prediction performance metrics.

Usage

```
sento_model(sentomeasures, y, x = NULL, ctr)
```

Arguments

`sentomeasures`	a `sentomeasures` object created using `sento_measures`. There should be at least two explanatory variables including the ones provided through the `x` argument.
`y`	a one-column `data.frame` or a `numeric` vector capturing the dependent (response) variable. In case of a logistic regression, the response variable is either a `factor` or a `matrix` with the factors represented by the columns as binary indicators, with the second factor level or column as the reference class in case of a binomial regression. No `NA` values are allowed.
`x`	a named `data.frame` with other explanatory variables as `numeric`, by default set to `NULL`.
`ctr`	output from a `ctr_model` call.

Details

Models are computed using the elastic net regularization as implemented in the **glmnet** package, to account for the multidimensionality of the sentiment measures. Additional explanatory variables are not subject to shrinkage. Independent variables are normalized in the regression process, but coefficients are returned in their original space. For a helpful introduction to **glmnet**, we refer to their vignette. The optimal elastic net parameters `lambda` and `alpha` are calibrated either through a to specify information criterion or through cross-validation (based on the "rolling forecasting origin" principle, using the `train` function). In the latter case, the training metric is automatically set to `"RMSE"` for a linear model and to `"Accuracy"` for a logistic model. We suppress many of the details that can be supplied to the `glmnet` and `train` functions we rely on, for the sake of user-friendliness.

Value

If ctr$do.iter = FALSE, a sentomodel object which is a list containing:

reg	optimized regression, i.e. a model-specific glmnet object.
model	the input argument ctr$model, to indicate the type of model estimated.
x	a matrix of the values used in the regression for all explanatory variables.
alpha	calibrated alpha.
lambda	calibrated lambda.
trained	output from train call (if ctr$type = "cv").
ic	a list composed of two elements: the information criterion used in the calibration under "criterion", and a vector of all minimum information criterion values for each value in alphas under "opts" (if ctr$type != "cv").
dates	sample reference dates as a two-element character vector, being the earliest and most recent date from the sentomeasures object accounted for in the estimation window.
nVar	the sum of the number of sentiment measures and other explanatory variables inputted.
discarded	a named logical vector of length equal to the number of sentiment measures, in which TRUE indicates that the particular sentiment measure has not been considered in the regression process. A sentiment measure is not considered when it is a duplicate of another, or when at least 25% of the observations are equal to zero.

If ctr$do.iter = TRUE, a sentomodeliter object which is a list containing:

models	all sparse regressions, i.e. separate sentomodel objects as above, as a list with as names the dates from the perspective of the sentiment measures at which predictions for performance measurement are carried out (i.e. one date step beyond the date sentomodel$dates[2]).
alphas	calibrated alphas.
lambdas	calibrated lambdas.
performance	a data.frame with performance-related measures, being "RMSFE" (root mean squared forecasting error), "MAD" (mean absolute deviation), "MDA" (mean directional accuracy, in which's calculation zero is considered as a positive; in percentage points), "accuracy" (proportion of correctly predicted classes in case of a logistic regression; in percentage points), and each's respective individual values in the sample. Directional accuracy is measured by comparing the change in the realized response with the change in the prediction between two consecutive time points (omitting the very first prediction, resulting in NA). Only the relevant performance statistics are given depending on the type of regression. Dates are as in the "models" output element, i.e. from the perspective of the sentiment measures.

Author(s)

Samuel Borms, Keven Bluteau

See Also

ctr_model, glmnet, train, retrieve_attributions

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
data("epu", package = "sentometrics")

# construct a sentomeasures object to start with
corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "2004-01-01" & date < "2014-10-01")
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
ctr <- ctr_agg(howWithin = "tf-idf", howDocs = "proportional",
               howTime = c("equal_weight", "linear"),
               by = "month", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

# prepare y and other x variables
y <- epu[epu$date >= sentomeasures$measures$date[1], ]$index
length(y) == nobs(sentomeasures) # TRUE
x <- data.frame(runif(length(y)), rnorm(length(y))) # two other (random) x variables
colnames(x) <- c("x1", "x2")

## Not run:
# a linear model based on the Akaike information criterion
ctrIC <- ctr_model(model = "gaussian", type = "AIC", do.iter = FALSE, h = 4,
                   do.difference = TRUE)
out1 <- sento_model(sentomeasures, y, x = x, ctr = ctrIC)

# attribution and prediction as post-analysis
attributions1 <- retrieve_attributions(out1, sentomeasures,
                                       refDates = sentomeasures$measures$date[20:25])
plot_attributions(attributions1, "features")

nx <- nmeasures(sentomeasures) + ncol(x)
newx <- runif(nx) * cbind(sentomeasures$measures[, -1], x)[30:40, ]
preds <- predict(out1, newx = as.matrix(newx), type = "link")
## End(Not run)

# an iterative out-of-sample analysis, parallelized
ctrIter <- ctr_model(model = "gaussian", type = "BIC", do.iter = TRUE, h = 3,
                     oos = 2, alphas = c(0.25, 0.75), nSample = 75, nCore = 2)
out2 <- sento_model(sentomeasures, y, x = x, ctr = ctrIter)
summary(out2)

## Not run:
# a cross-validation based model, parallelised
cl <- parallel::makeCluster(2)
doParallel::registerDoParallel(cl)
ctrCV <- ctr_model(model = "gaussian", type = "cv", do.iter = FALSE,
```

```
                    h = 0, alphas = c(0.10, 0.50, 0.90), trainWindow = 70,
                    testWindow = 10, oos = 0, do.progress = TRUE)
out3 <- sento_model(sentomeasures, y, x = x, ctr = ctrCV)
parallel::stopCluster(cl)
summary(out3)

# a cross-validation based model for a binomial target
yb <- epu[epu$date >= sentomeasures$measures$date[1], ]$above
ctrCVb <- ctr_model(model = "binomial", type = "cv", do.iter = FALSE,
                    h = 0, alphas = c(0.10, 0.50, 0.90), trainWindow = 70,
                    testWindow = 10, oos = 0, do.progress = TRUE)
out4 <- sento_model(sentomeasures, yb, x = x, ctr = ctrCVb)
summary(out4)
## End(Not run)
```

`setup_lexicons` *Set up lexicons (and valence word list) for use in sentiment analysis*

Description

Structures provided lexicons and potentially integrates valence words. One can also provide (part of) the built-in lexicons from `data("list_lexicons")` or a valence word list from `data("list_valence_shifters")` as an argument. Part of this function mimicks the `as_key` function from the **sentimentr** package to make the output coherent, convert all words to lowercase and check for duplicates.

Usage

```
setup_lexicons(lexiconsIn, valenceIn = NULL, do.split = FALSE)
```

Arguments

`lexiconsIn`	a named `list` of (raw) lexicons, each element as a `data.frame` or a `data.table` with respectively a words column and a polarity score column. Alternatively, a subset of the already formatted built-in lexicons accessible via `list_lexicons` can be declared too, as part of the same list input. If only (some of) the package built-in lexicons want to be used (with *no* valence shifters), one can simply supply `list_lexicons[c(...)]` as an argument to either `sento_measures` or `compute_sentiment`. However, it is strongly recommended to pass all lexicons (and a valence word list) to this function first, in any case.
`valenceIn`	a single valence word list as a `data.frame` or a `data.table` with respectively a words column, a type column (`1` for negators, `2` for amplifiers/intensifiers, and `3` for deamplifiers/downtoners) and a score column. The scores should be the same within each type. This argument can be one of the already formatted built-in valence word lists accessible via `list_valence_shifters`. If `NULL`, no valence word list is part of this function's output, nor will it applied in the sentiment analysis.
`do.split`	a `logical` that if `TRUE` splits every lexicon into a separate positive polarity and negative polarity lexicon.

Value

A `list` with each lexicon as a separate element according to its name, as a `data.table`, and optionally an element named `valence` that comprises the valence words. Every `x` column contains the words, every `y` column contains the polarity score, and for the valence word list, `t` contains the word type. If a valence word list is provided, all lexicons are expanded by copying the respective lexicon, and changing the words and scores according to the valence word type: "NOT_" is added for negators, "VERY_" is added for amplifiers and "HARDLY_" is added for deamplifiers. New lexicon scores are obtained by multiplication of the original lexicon scores with the first value of the scores column of the valence word list, per type (thus why valence scores should be the same across the types).

Author(s)

Samuel Borms

Examples

```
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# lexicons straight from built-in word lists
l1 <- list_lexicons[c("LM_en", "HENRY_en")]

# including a self-made lexicon, with and without valence shifters
lexIn <- c(list(myLexicon = data.table(w = c("nice", "boring"), s = c(2, -1))),
           list_lexicons[c("GI_en")])
valIn <- list_valence_shifters[["en"]]
l2 <- setup_lexicons(lexIn)
l3 <- setup_lexicons(lexIn, valIn)
l4 <- setup_lexicons(lexIn, valIn, do.split = TRUE)

## Not run:
# include lexicons from lexicon package
lexIn2 <- list(hul = lexicon::hash_sentiment_huliu, joc = lexicon::hash_sentiment_jockers)
l5 <- setup_lexicons(c(lexIn, lexIn2), valIn)
## End(Not run)
```

---

to_global *Merge sentiment measures into multiple weighted global sentiment indices*

Description

Merges all sentiment measures into a weighted global textual sentiment measure for each of the `lexicons`, `features`, and `time` dimensions.

Usage

```
to_global(sentomeasures, lexicons = 1, features = 1, time = 1)
```

Arguments

sentomeasures	a sentomeasures object created using sento_measures.
lexicons	a numeric vector of weights, of size length(sentomeasures$lexicons), in the same order. By default set to 1, which means equally weighted.
features	a numeric vector of weights, of size length(sentomeasures$features), in the same order. By default set to 1, which means equally weighted.
time	a numeric vector of weights, of size length(sentomeasures$time), in the same order. By default set to 1, which means equally weighted.

Details

This function returns no new sentomeasures object. The global sentiment measures as outputted can still easily be added to regressions as an additional variable using the x argument in the sento_model function. The measures are constructed from weights that indicate the importance (and sign) along each component from the lexicons, features, and time dimensions. There is no condition in terms of allowed weights. For example, the global index based on the supplied lexicon weights ("globLex") is obtained first by multiplying every sentiment measure with its corresponding weight (meaning, the weight given to the lexicon the sentiment is computed with), then by taking the average per date.

Value

A data.frame with the different types of weighted global sentiment measures, named "globLex", "globFeat", "globTime" and "global", with dates as row names. The last measure is an average of the the three other measures.

Author(s)

Samuel Borms

See Also

sento_model

Examples

```
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- setup_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
```

```
sentomeasures <- sento_measures(corpusSample, l, ctr)

# merge into one global sentiment measure, with specified weighting for lexicons and features
global <- to_global(sentomeasures, lexicons = c(0.40, 0.60),
                                   features = c(0.10, -0.20, 0.30, -1),
                                   time = 1)
```

`to_sentocorpus` *Convert a quanteda corpus object into a sentocorpus object*

Description

Converts a `corpus` object into a `sentocorpus` object, by adding user-supplied dates and doing proper rearranging.

Usage

```
to_sentocorpus(corpus, dates, do.clean = FALSE)
```

Arguments

`corpus`	a quanteda `corpus` object.
`dates`	a sequence of dates as `"yyyy-mm-dd"`, of the same length as the number of documents in the input `corpus`.
`do.clean`	see `do.clean` argument from the `sento_corpus` function.

Value

A `sentocorpus` object, as returned by the `sento_corpus` function.

Author(s)

Samuel Borms

See Also

`corpus`, `sento_corpus`

Examples

```
data("usnews", package = "sentometrics")

# reshuffle usnews data.frame
dates <- usnews$date
usnews$id <- usnews$date <- NULL
usnews$wrong <- "notNumeric"
colnames(usnews)[1] <- "myTexts"
```

```
# set up quanteda corpus object
corpusQ <- quanteda::corpus(usnews, text_field = "myTexts")

# corpus conversion
corpusS <- to_sentocorpus(corpusQ, dates = dates)
```

---

usnews *Texts (not) relevant to the U.S. economy*

Description

A collection of texts annotated by humans in terms of relevance to the U.S. economy or not. The texts come from two major journals in the U.S. (The Wall Street Journal and The Washington Post) and cover 4145 documents between 1995 and 2014. It contains following information:

- id. A `character` ID identifier.
- date. Date as `"yyyy-mm-dd"`.
- texts. Texts in `character` format.
- wsj. Equals 1 if the article comes from The Wall Street Journal.
- wapo. Equals 1 if the article comes from The Washington Post (complementary to 'wsj').
- economy. Equals 1 if the article is relevant to the U.S. economy.
- noneconomy. Equals 1 if the article is not relevant to the U.S. economy (complementary to 'economy').

Usage

```
data("usnews")
```

Format

A `data.frame`, formatted as required to be an input for `sento_corpus`.

Source

Economic News Article Tone and Relevance

Examples

```
data("usnews", package = "sentometrics")
usnews[3192, "texts"]
usnews[1:5, c("id", "date", "texts")]
```

Index