Package ‘sentometrics’

December 18, 2018

Type Package

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

Version 0.5.6

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. (2018) <doi:10.2139/ssrn.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 covr, doParallel, e1071, parallel, randomForest, testthat

Imports caret, compiler, foreach, ggplot2, glmnet, ISOweek, quanteda,
Rcpp (>= 0.12.13), RcppRoll, RcppParallel, stats, stringi,
utils

LinkingTo Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 6.1.0

SystemRequirements GNU make

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-12-18 12:10:03 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentometrics-package</td>
<td>3</td>
</tr>
<tr>
<td>add_features</td>
<td>4</td>
</tr>
<tr>
<td>aggregate.sentiment</td>
<td>6</td>
</tr>
<tr>
<td>attributions</td>
<td>7</td>
</tr>
<tr>
<td>compute_sentiment</td>
<td>8</td>
</tr>
<tr>
<td>ctr_agg</td>
<td>10</td>
</tr>
<tr>
<td>ctr_model</td>
<td>12</td>
</tr>
<tr>
<td>diff.sentomeasures</td>
<td>15</td>
</tr>
<tr>
<td>epu</td>
<td>16</td>
</tr>
<tr>
<td>get_dates</td>
<td>17</td>
</tr>
<tr>
<td>get_dimensions</td>
<td>17</td>
</tr>
<tr>
<td>get_hows</td>
<td>18</td>
</tr>
<tr>
<td>get_loss_data</td>
<td>18</td>
</tr>
<tr>
<td>get_measures</td>
<td>20</td>
</tr>
<tr>
<td>list_lexicons</td>
<td>21</td>
</tr>
<tr>
<td>list_valence_shifters</td>
<td>22</td>
</tr>
<tr>
<td>measures_delete</td>
<td>23</td>
</tr>
<tr>
<td>measures_fill</td>
<td>24</td>
</tr>
<tr>
<td>measures_global</td>
<td>25</td>
</tr>
<tr>
<td>measures_merge</td>
<td>26</td>
</tr>
<tr>
<td>measures_select</td>
<td>28</td>
</tr>
<tr>
<td>measures_subset</td>
<td>29</td>
</tr>
<tr>
<td>nmeasures</td>
<td>30</td>
</tr>
<tr>
<td>peakdates</td>
<td>31</td>
</tr>
<tr>
<td>peakdocs</td>
<td>32</td>
</tr>
<tr>
<td>plot.attributions</td>
<td>33</td>
</tr>
<tr>
<td>plot.sentomeasures</td>
<td>34</td>
</tr>
<tr>
<td>plot.sentomodeliter</td>
<td>35</td>
</tr>
<tr>
<td>predict.sentomodel</td>
<td>36</td>
</tr>
<tr>
<td>scale.sentomeasures</td>
<td>37</td>
</tr>
<tr>
<td>sentiment_bind</td>
<td>38</td>
</tr>
<tr>
<td>sento_corpus</td>
<td>39</td>
</tr>
<tr>
<td>sento_lexicons</td>
<td>41</td>
</tr>
<tr>
<td>sento_measures</td>
<td>42</td>
</tr>
<tr>
<td>sento_model</td>
<td>44</td>
</tr>
<tr>
<td>to_sentiment</td>
<td>47</td>
</tr>
<tr>
<td>to_sentocorpus</td>
<td>48</td>
</tr>
<tr>
<td>usnews</td>
<td>49</td>
</tr>
<tr>
<td>weights_almon</td>
<td>50</td>
</tr>
<tr>
<td>weights_beta</td>
<td>51</td>
</tr>
<tr>
<td>weights_exponential</td>
<td>52</td>
</tr>
</tbody>
</table>

Index 53
sentometrics-package

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

Description

The sentometrics package is an integrated framework for textual sentiment time series aggregation and prediction. It accounts for the intrinsic challenge that, for a given text, sentiment can be computed in many different ways, as well as the large number of possibilities to pool sentiment across texts and time. This additional layer of manipulation does not exist in standard text mining and time series analysis packages. The package therefore integrates the fast qualification of sentiment from texts, the aggregation into different sentiment time series and the optimized prediction based on these measures.

Main functions

- Corpus features generation: sento_corpus, add_features
- Sentiment computation and aggregation into sentiment measures: ctr_agg, sento_lexicons, compute_sentiment, aggregate.sentiment, sento_measures, peakdocs, peakdates, and a series of measures_xyz, generic and extractor functions
- Sparse modelling: ctr_model, sento_model
- Prediction and post-modelling analysis: predict.sentomodel, attributions

Update

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

Note

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>

References


See Also

Useful links:

- [https://github.com/sborms/sentometrics](https://github.com/sborms/sentometrics)
- Report bugs at [https://github.com/sborms/sentometrics/issues](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 a `quanteda` corpus object.

**Usage**

```r
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 column 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 just keywords, one can also directly use a single regex expression to define a new feature (see the details section).
- **do.binary**: a logical, if `do.binary = FALSE`, the number of occurrences are normalized between 0 and 1 (see argument `keywords`).
- **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.
add_features

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. We use the stringi package for searching the keywords. 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 this transformation, and are evaluated as given, if the corresponding keywords vector consists of only one expression. For a large corpus and/or complex regex patterns, this function may require some patience. Scaling between 0 and 1 is performed via min-max normalization, per column.

Value

An updated corpus object.

Author(s)

Samuel Borms

Examples

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

set.seed(505)

# construct a corpus and add (a) feature(s) to it
corpus <- quanteda::corpus_sample(sento_corpus(corpusdf = usnews), 500)
corpus1 <- add_features(corpus,
    featuresdf = data.frame(random = runif(quanteda::ndoc(corpus))))
corpus2 <- add_features(corpus,
    keywords = list(pres = "president", war = "war"),
    do.binary = FALSE)
corpus3 <- add_features(corpus,
    keywords = list(pres = c("Obama", "US president")))
corpus4 <- add_features(corpus,
    featuresdf = data.frame(all = 1),
    keywords = list(pres1 = "Obama|US [pP]resident",
                   pres2 = "\bObama\b\bUS president\b",
                   war = "war"),
    do.regex = c(TRUE, TRUE, FALSE))

sum(corpus$documents$pres) == sum(corpus$documents$pres2) # TRUE

# adding a complementary feature
nonpres <- data.frame(nonpres = as.numeric(!quanteda::docvars(corpus3)[["pres"]]))
corpus3 <- add_features(corpus3,
    featuresdf = nonpres)
aggregate.sentiment  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

```r
## S3 method for class 'sentiment'
aggregate(x, ctr, ...)
```

Arguments

- `x`: a sentiment object created using compute_sentiment (from a sento_corpus object), or an output from to_sentiment.
- `ctr`: output from a ctr_agg call. The howWithin and nCore elements are ignored.
- `...`: not used.

Value

A sento_measures object.

Author(s)

Samuel Borms, Keven Bluteau

See Also

compute_sentiment, ctr_agg, sento_measures

Examples

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

corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- sento_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 <- aggregate(sent, ctr)
```
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

attributions(model, sentomeasures, do.lags = TRUE,
  do.normalize = FALSE, refDates = NULL, factor = NULL)

Arguments

model a sentomodel or a sentomodeliter object created with sento_model.
sentomeasures the sentomeasures object, as created with sento_measures, used to estimate the model from the first argument (make sure this is the case!).
do.lags a logical, TRUE also computes the attribution to each time lag. For large time lags, this is time-consuming.
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 (that is, model$dates[1] if model is a sentomodel object). All dates should also be in get_dates(sentomeasures). 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. A NULL value for document-level attribution on a given date means no documents are directly implicated in the associated prediction.
compute_sentiment

Value
A list of class attributions, with "documents", "lags", "lexicons", "features" and "time" as dimensions for which aggregation is computed. The last four 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. If do.lags = FALSE, the "lags" element is set to NULL.

Author(s)
Samuel Borms, Keven Bluteau

See Also
sento_model

compute_sentiment  Compute document-level sentiment across features and lexicons

Description
Given a corpus of texts, computes (net) sentiment per document using the bag-of-words approach based on the lexicons provided and a choice of aggregation across words per document.

Usage
compute_sentiment(x, lexicons, how = "proportional", tokens = NULL, nCore = 1)

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 a sentolexicons object created using sento_lexicons.

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.

tokens a list of tokenized documents, to specify your own tokenization scheme. Can result from the quanteda's tokens function, the tokenizers package, or other. Make sure the tokens are constructed from (the texts from) the x argument, are unigrams, and preferably set to lowercase, otherwise, results may be spurious and errors could occur. By default set to NULL.
**compute_sentiment**

nCore: a positive numeric that will be passed on to the numThreads argument of the setThreadOptions function, to parallelize the sentiment computation across texts. A value of 1 (default) implies no parallelization. Parallelization is expected to improve speed of the sentiment computation only for sufficiently large corpora.

**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 sento_lexicons function, which splits out the lexicons into a positive and a negative polarity counterpart. All NAs are converted to 0, under the assumption that this is equivalent to no sentiment. If tokens = NULL (as per default), texts are tokenized as unigrams using the tokenize_words function. Punctuation and numbers are removed, but not stopwords. The number of words for each document is computed based on that same tokenization. All tokens are converted to lowercase, in line with what the sento_lexicons function does for the lexicons and valence shifters.

**Value**

If x is a sentocorpus object, a sentiment object, i.e., a data.table containing the sentiment scores data.table with an "id", a "date" and a "word_count" column, and all lexicon–feature sentiment scores columns. A sentiment object can be used for aggregation into time series with the aggregate.sentiment function.

If x is a quanteda corpus object, a sentiment scores data.table with an "id" and a "word_count" column, and all lexicon–feature sentiment scores columns.

If x is a character vector, a sentiment scores data.table with a "word_count" column, and all lexicon–feature sentiment scores columns.

**Calculation**

If the lexicons argument has no "valence" element, the sentiment computed corresponds to simple unigram matching with the lexicons [unigrams approach]. If valence shifters are included in lexicons with a corresponding "y" column, these have the effect of modifying the polarity of a word detected from the lexicon if appearing right before such word (examples: not good, very bad or can’t defend) [bigrams approach]. If the valence table contains a "t" column, valence shifters are searched for in a cluster centered around a detected polarity word [clusters approach]. The latter approach is similar along the one utilized by the sentimentr package, but simplified. A cluster amounts to four words before and two words after a polarity word. A cluster never overlaps with a preceding one. Roughly speaking, the polarity of a cluster is calculated as $n(1 + 0.80d)S + \sum s$. The polarity score of the detected word is $S$, $s$ represents polarities of eventual other sentiment words, and $d$ is the difference between the number of amplifiers ($t = 2$) and the number of deamplifiers ($t = 3$). If there is an odd number of negators ($t = 1$), $n = -1$ and amplifiers are counted as deamplifiers, else $n = 1$. All scores, whether per unigram, per bigram or per cluster, are summed within a document, before the scaling defined by the how argument is applied. The how = "proportionalPol" option divides each document’s sentiment score by the number of detected polarized words (counting words that appear multiple times by their frequency), instead of the total number of words which the how = "counts" option does no normalization. See the vignette for more details.
Author(s)

Samuel Borms

Examples

```r
# from a sentocorpus object, unigrams approach
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 200)
sent1 <- compute_sentiment(corpusSample, ll, how = "proportionalPol")

# from a character vector, bigrams approach
sent2 <- compute_sentiment(usnews["texts"][1:200], l2, how = "counts")

# from a corpus object, clusters approach
corpusQ <- quanteda::corpus(usnews, text_field = "texts")
corpusQSample <- quanteda::corpus_sample(corpusQ, size = 200)
sent3 <- compute_sentiment(corpusQSample, l3, how = "counts")

# from an already tokenized corpus, using the 'tokens' argument
toks <- as.list(quanteda::tokens(corpusQSample, what = "fastestword"))
sent4 <- compute_sentiment(corpusQSample, ll[1], how = "counts", tokens = toks)
```

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

```r
ctr_agg(whatWithin = "proportional", howDocs = "equal_weight", howTime = "equal_weight", do.ignoreZeros = TRUE, by = "day", lag = 1, fill = "zero", alphasExp = seq(0.1, 0.5, by = 0.1), ordersAIm = 1:3, do.inverseAIm = TRUE, aBeta = 1:4, bBeta = 1:4, weights = NULL, tokens = NULL, nCore = 1)
```
Arguments

howWithin  a single character vector defining how aggregation within documents will be performed. Should length(howWithin) > 1, the first element is used. For available options on how this 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 available options on how this 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 available options on how this 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 1, meaning no aggregation across time; a time weighting scheme named "dummyTime" is used in this case.

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); see weights_exponential.

ordersAlm  a numeric vector of all Almon polynomial orders (positive) to calculate weights for, used if "almon" %in% howTime; see weights_almon.

do.inverseAlm  a logical indicating if for every Almon polynomial its inverse has to be added, used if "almon" %in% howTime; see weights_almon.

aBeta  a numeric vector of positive values as first Beta weighting decay parameter; see weights_beta.

bBeta  a numeric vector of positive values as second Beta weighting decay parameter; see weights_beta.

weights  optional own weighting scheme(s), used if provided as a data.frame with the number of rows equal to the desired lag.

tokens  see compute_sentiment.

ncore  see compute_sentiment.
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

set.seed(505)

# 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", "beta", "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),
               aBeta = c(1, 3),
               bBeta = c(1, 3, 4, 7),
               weights = data.frame(myWeights = runif(20)))

# set up control function with one linear and two chosen Almon weighting schemes
a <- weights_almon(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_model

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

```r
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), lambdas = NULL, nSample = NULL,
         trainWindow = NULL, testWindow = NULL, start = 1,
         do.shrinkage.x = FALSE, 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 (Tibshirani and Taylor, 2012; Zou, Hastie and Tibshirani, 2007), and "cv" (cross-validation based on the `train` function from the `caret` package). The adapted information criteria are 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 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-sample 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, but 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 yt+2 − yt on Xt. If h = -2, the regression fitted is yt+2 − yt on Xt-2. The argument is always kept at FALSE if the model argument is one of c("binomial", "multinomial").
- **alphas**: a numeric vector of the 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.
lambdas a numeric vector of the lambdas to test for during calibration, greater or equal than zero. A value of zero means no regularization, thus requires care when the data is fat. By default set to NULL, such that the lambdas sequence is generated 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.shrinkage.x a logical vector to indicate which of the other regressors provided through the x argument of the sento_model function should be subject to shrinkage (TRUE). If argument is of length one, it applies to all external regressors.

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

ncore a positive integer 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

References


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)

---

diff.sentomeasures  

**Differencing of sentiment measures**

**Description**

Differences the sentiment measures from a sentomeasures object.

**Usage**

```r
## 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)
1 <- sento_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, 1, 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 news-based U.S. Economic Policy Uncertainty (EPU) index (Baker, Bloom and Davis, 2015). Goes from January 1985 to July 2018, and includes a binomial and a multinomial example series. Following columns are present:

- **date.** Date as "yyy-mm-dd".
- **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 403 rows and 4 columns.

**Source**

get_dates

References

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

get_dates
Get the dates of the sentiment measures/time series

Description
Returns the dates of the sentiment time series.

Usage
get_dates(sentomeasures)

Arguments
sentomeasures a sentomeasures object created using sento_measures.

Value
The "date" column in sentomeasures[[“measures”]] as a character vector.

Author(s)
Samuel Borms

get_dimensions
Get the dimensions of the sentiment measures

Description
Returns the components across all three dimensions of the sentiment measures.

Usage
get_dimensions(sentomeasures)

Arguments
sentomeasures a sentomeasures object created using sento_measures.
Value

The "features", "lexicons" and "time" elements in sentomeasures.

Author(s)

Samuel Borms

get_hows

Options supported to perform aggregation into sentiment measures

Description

Outputs the supported aggregation arguments. 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 detailed explanation of all aggregation options.

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

get_loss_data

Retrieve loss data from a selection of models

Description

Structures specific performance data for a set of different sentomodeliter objects as loss data. Can then be used, for instance, as an input to create a model confidence set (Hansen, Lunde and Nason, 2011) with the MCS package.

Usage

get_loss_data(models, loss = c("DA", "error", "errorSq", "AD", "accuracy"))
get_loss_data

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 inaccuracy), "error" (prediction minus realized response variable), "errorSq" (squared errors), "AD" (absolute errors) or "accuracy" (inaccurate class predictions). This argument defines on what basis the model confidence set is calculated. The first four options are available for "gaussian" models, the last option applies only to "binomial" and "multinomial" models.

Value

A matrix of loss data.

Author(s)

Samuel Borms

References


See Also

sentomodel, MCSprocedure

Examples

```r
## Not run:
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
data("epu", package = "sentometrics")
set.seed(505)

# construct two sentomeasures objects
corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "1997-01-01" & date < "2014-10-01")
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctrA <- ctr_agg(howWithin = "proportionalPol", howDocs = "proportional",
               howTime = c("equal_weight", "linear"), by = "month", lag = 3)
sentMeas <- sento_measures(corpus, l, ctrA)

# prepare y and other x variables
y <- epu[epu$date %in% get_dates(sentMeas), "index"]
length(y) == nobs(sentMeas) # 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
ctrM <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
   h = 0, nSample = 120, start = 50)
out1 <- sento_model(sentMeas, y, x = x, ctr = ctrM)
out2 <- sento_model(sentMeas, y, x = NULL, ctr = ctrM)
out3 <- sento_model(measures_select(sentMeas, "linear"), y, x = x, ctr = ctrM)
out4 <- sento_model(measures_select(sentMeas, "linear"), y, x = NULL, ctr = ctrM)

lossData <- get_loss_data(models = list(m1 = out1, m2 = out2, m3 = out3, m4 = out4),
   loss = "errorSq")
mcs <- MCS::MCSprocedure(lossData)
## End(Not run)

---

get_measures  Get the sentiment measures

Description

Returns the sentiment measures in either wide (by default) or long format.

Usage

get_measures(sentomeasures, format = "wide")

Arguments

sentomeasures  a sentomeasures object created using sento_measures.
format  a single character vector, one of c("wide", "long").

Value

The panel of sentiment measures under sentomeasures["measures"], in wide or long format.

Author(s)

Samuel Borms
**Description**

A list containing all built-in lexicons as a data.frame 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 unigrams and in lowercase. The built-in lexicons are the following:

- `FEEL_en_tr`
- `FEEL_fr` (Abdaoui, Azé, Bringay and Poncelet, 2017)
- `FEEL_nl_tr`
- `GI_en` (General Inquirer, i.e. Harvard IV-4 combined with Laswell)
- `GI_fr_tr`
- `GI_nl_tr`
- `HENRY_en` (Henry, 2008)
- `HENRY_fr_tr`
- `HENRY_nl_tr`
- `LM_en` (Loughran and McDonald, 2011)
- `LM_fr_tr`
- `LM_nl_tr`

Other useful lexicons 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**

References


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, as data.tables with three columns: a x column with the words, a y column with the values associated to each word, and a t column with the type of valence shifter (1 = negators, 2 = amplifiers, 3 = deamplifiers). 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. All words are unigrams and 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.

Source


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)

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, or as a list of character vectors, possibly with separately specified combinations (consisting of one unique lexicon, one unique feature, and one unique 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 <- quantdata::corpus_sample(corpus, size = 500)
l <- sento_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("linear", "LM_en"))
measures_fill

Description

Adds missing dates between earliest and latest date of a sentomeasures object or two more extreme boundary dates, such that the time series are continuous date-wise. Fills in any missing date with either 0 or the most recent non-missing value.

Usage

```r
measures_fill(sentomeasures, fill = "zero", dateBefore = NULL, dateAfter = NULL)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentomeasures</td>
<td>a sentomeasures object created using <code>sento_measures</code>.</td>
</tr>
<tr>
<td>fill</td>
<td>an element of <code>c(&quot;zero&quot;, &quot;latest&quot;)</code>; the first assumes missing dates represent zero sentiment, the second assumes missing dates represent constant sentiment.</td>
</tr>
<tr>
<td>dateBefore</td>
<td>a date as &quot;yyyy-mm-dd&quot;, to stretch the sentiment time series from up to the first date. Should be earlier than <code>get_dates(sentomeasures)[1]</code> to take effect. The values for these dates are set to those at <code>get_dates(sentomeasures)[1]</code>. If <code>NULL</code>, then ignored.</td>
</tr>
<tr>
<td>dateAfter</td>
<td>a date as &quot;yyyy-mm-dd&quot;, to stretch the sentiment time series up to this date. Should be later than <code>tail(get_dates(sentomeasures), 1)</code> to take effect. If <code>NULL</code>, then ignored.</td>
</tr>
</tbody>
</table>

Details

The `dateBefore` and `dateAfter` dates are converted according to the `sentomeasures[["by"]]]` frequency.

Value

A modified sentomeasures object.

Author(s)

Samuel Borms
measures_global

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 <- sento.lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "day", lag = 7, fill = "none")
sentomeasures <- sento_measures(corpusSample, l, ctr)

# fill measures
f1 <- measures_fill(sentomeasures)
f2 <- measures_fill(sentomeasures, fill = "latest")
f3 <- measures_fill(sentomeasures, fill = "zero",
  dateBefore = get_dates(sentomeasures)[1] - 10,
  dateAfter = tail(get_dates(sentomeasures), 1) + 15)

measures_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

measures_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

In contrast to other measures_xyz functions, this particular function returns no new sentomeasures object. 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 restriction 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.table with the different types of weighted global sentiment measures, named "globLex", "globFeat", "globTime" and "global", with "date" as the first column. 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 <- sento_lexicons(list_lexicons[["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 <- measures_global(sentomeasures,
  lexicons = c(0.40, 0.60),
  features = c(0.10, -0.20, 0.30, -1),
  time = 1)
Description

Merges sentiment 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(sentomeasures, features = NULL, lexicons = NULL, 
    time = NULL, 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$features` for the exact names to use. Use NULL (default) to apply no merging across this dimension.
- **lexicons**: a list with unique lexicons to merge at given name, e.g., `list(lex12 = c("lex1", "lex2"))`. See `sentomeasures$lexicons` for the exact names to use. Use NULL (default) 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 NULL (default) 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 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

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 <- sento_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)
```
# merging across specified components
sentomeasuresMerged <- measures_merge(sentomeasures,
  time = list(W = c("equal_weight", "linear")),
  features = list(journals = c("wsj", "wapo")),
  do.keep = TRUE)

# merging in full
dims <- get_dimensions(sentomeasures)
sentomeasuresFull <- measures_merge(sentomeasures,
  lexicons = list(L = dims[["lexicons"]]),
  time = list(T = dims[["time"]]),
  features = list(F = dims[["features"]]))

## Not run:
# this merging will not work, but produces an informative error message
measures_merge(sentomeasures,
  time = list(W = c("equal_weight", "almon")),
  lexicons = list(LEX = c("LM_en")),
  features = list(journals = c("notInHere", "wapo")))
## End(Not run)

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)

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, or as a list of character vectors,
possibly with separately specified combinations (consisting of one unique lexi-
con, one unique feature, and one unique 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
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 <- sento.lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(time = 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
peakdates

Extract dates related to sentiment time series peaks

Description
This function extracts the dates 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

peakdates(sentomeasures, n = 10, type = "both", do.average = FALSE)

Arguments

sentomeasures a sentomeasures object created using sento_measures.
n a positive numeric value to indicate the number of dates associated to sentiment peaks to extract. If n < 1, it is interpreted as a quantile (for example, 0.07 would mean the 7% most extreme dates).
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 vector of type "Date" corresponding to the n extracted sentiment peak dates.

Author(s)
Samuel Borms

Examples

set.seed(505)

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 <- sento_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)
peakdocs

Extract documents related to sentiment peaks

Description

This function extracts the documents with most extreme sentiment (lowest, highest or both in absolute terms). The extracted documents are unique, even when, for example, all most extreme sentiment values (across sentiment calculation methods) occur only for one document.

Usage

peakdocs(sentiment, n = 10, type = "both", do.average = FALSE)

Arguments

sentiment a sentiment object created using compute_sentiment or to_sentiment.
n a positive numeric value to indicate the number of dates associated to sentiment peaks to extract. If \( n < 1 \), it is interpreted as a quantile (for example, 0.07 would mean the 7% most extreme dates).
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 vector of type "character" corresponding to the \( n \) extracted document identifiers.

Author(s)

Samuel Borms

Examples

```
set.seed(505)

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

1 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])

corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 200)
sent <- compute_sentiment(corpusSample, 1, how = "proportionalPol")

# extract the peaks
peaksAbs <- peakdocs(sent, n = 5)
peaksAbsQuantile <- peakdocs(sent, n = 0.50)
peaksPos <- peakdocs(sent, n = 5, type = "pos")
peaksNeg <- peakdocs(sent, n = 5, type = "neg")

plot.attributions

### Plot prediction attributions at specified level

**Description**

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

**Usage**

```r
## S3 method for class 'attributions'
plot(x, group = "features", ...)
```

**Arguments**

- `x`: an attributions object created with `attributions`.
- `group`: a value from `c("lags", "lexicons", "features", "time")`. Not used.

**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. By default, a legend is positioned at the top if the number of components of the dimension is at maximum twelve.

**Author(s)**

Samuel Borms, Keven Bluteau
plot.sentomeasures

Description
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 a measures_xyz function when you want to plot only a subset of the sentiment measures.

Usage

```r
## S3 method for class 'sentomeasures'
plot(x, group = "all", ...) # 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 example). By default, a legend is positioned at the top if there are at maximum twelve line graphs plotted and group is different from "all".

Author(s)
Samuel Borms

Examples
```r
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 <- sento_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.sentomodeliter

plot(sentomeasures, group = "features")

## Not run:
# adjust appearance of plot
p <- plot(sentomeasures)
p <- p +
  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.

Details

See sento_model for an elaborate modelling example including the plotting of out-of-sample performance.

Value

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

Author(s)

Samuel Borms
Description

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

Usage

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

Arguments

- `object`: a `sentomodel` object created with `sento_model`.
- `newx`: a data matrix used for the prediction(s), row-by-row; see `predict.glmnet`. The number of columns should be equal to `sum(sentomodel$nVar)`, being the number of original sentiment measures and other variables. The variables discarded in the regression process are dealt with within this function, based on `sentomodel$discarded`.
- `type`: type of prediction required, a value from `c("link", "response", "class")`, see documentation for `predict.glmnet`.
- `offset`: not used.
- `...`: not used.

Value

A prediction output depending on the `type` argument.

Author(s)

Samuel Borms

See Also

`predict.glmnet`, `sento_model`
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

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

Arguments

- `x`: a `sentomeasures` object created using `sento_measures`.
- `center`: a logical or a numeric vector; see documentation for the generic `scale`. Alternatively, one can provide a matrix of dimensions `nobs(sentomeasures)` times 1 or `nmeasures(sentomeasures)` with values to add to each individual observation.
- `scale`: a logical or a numeric vector, see documentation for the generic `scale`. Alternatively, one can provide a matrix of dimensions `nobs(sentomeasures)` times 1 or `nmeasures(sentomeasures)` with values to divide each individual observation by.

Details

If one of the arguments `center` or `scale` is a matrix, this operation will be applied first, and eventual other centering or scaling is computed on that data.

Value

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

Author(s)

Samuel Borms

Examples

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

set.seed(505)

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- sento_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 to zero mean and unit standard deviation
sc1 <- scale(sentomeasures)

n <- nobs(sentomeasures)
m <- nmeasures(sentomeasures)

# add a matrix
sc2 <- scale(sentomeasures, center = matrix(runif(n * m), n, m), scale = FALSE)

# divide every row observation based on a one-column matrix, then center
sc3 <- scale(sentomeasures, center = TRUE, scale = matrix(runif(n)))

sentiment_bind

Bind sentiment objects row-wise

Description
Combines multiple sentiment objects with the same column names into a new sentiment object. Duplicates in terms of document identifiers across input objects are removed.

Usage
sentiment_bind(...)

Arguments
...          sentiment objects to combine in the order given.

Value
A new, larger, sentiment object.

Author(s)
Samuel Borms

Examples
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
```r
corp1 <- sento_corpus(corpusdf = usnews[1:200, ])
corp2 <- sento_corpus(corpusdf = usnews[201:450, ])
corp3 <- sento_corpus(corpusdf = usnews[401:700, ])

sent1 <- compute_sentiment(corp1, l, how = "proportionalPol")
sent2 <- compute_sentiment(corp2, l, how = "counts")
sent3 <- compute_sentiment(corp3, l, how = "proportional")

sent <- sentiment_bind(sent1, sent2, sent3)
nrow(sent) # 700
```

---

### sento_corpus

Create a sentocorpus object

---

**Description**

Formalizes a collection of texts into a well-defined corpus object derived from the `corpus` object. The `quanteda` package provides a robust text mining infrastructure, see `quanteda`. Their corpus structure brings for example a handy corpus manipulation toolset. This function performs a set of checks on the input data and prepares the corpus for further 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 (in character mode), a "date" column (as "yyyy-mm-dd"), a "texts" column (in character mode), and a series of feature columns of type numeric, with values between 0 and 1 to specify the degree of connectedness of a feature to a document. Features could be topics (e.g., legal or economic), article sources (e.g., online or print), amongst many more options. When no feature column is provided, a feature named "dummyFeature" is added. All spaces in the names of the features are replaced by `_`. Feature columns with values not between 0 and 1 are rescaled column-wise.

- `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 quantedan corpus. Any quantedan function applicable to its corpus object can also be applied to a sentocorpus object. However, changing a given sentocorpus object too drastically using some of quantedan'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. Binary features are useful as a mechanism to select the texts which have to be integrated in the respective feature-based sentiment measure(s), but applies only when do.ignoreZeros = TRUE. Because of this (implicit) selection that can be performed, having complementary features (e.g., "economy" and "noneconomy") makes sense.

Value

A sentocorpus object, derived from a quantedan corpus classed list with 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
corp <- sento_corpus(corpusdf = usnews)

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

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

# deleting all features results in the addition of a dummy feature
quantedan::docvars(corp, field = c("economy", "noneconomy", "wsj")) <- NULL

## Not run:
# to add or replace features, use the add_features() function...
quantedan::docvars(corp, field = c("wsj", "new")) <- 1
## End(Not run)

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

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

Description

Structures provided lexicon(s) and optionally valence words. One can for example combine (part of) the built-in lexicons from data("list_lexicons") with other lexicons, and add one of the built-in valence word lists from data("list_valence_shifters"). This function makes the output coherent, by converting all words to lowercase and checking for duplicates. All entries consisting of more than one word are discarded, as required for bag-of-words sentiment analysis.

Usage

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

Arguments

- **lexiconsIn**: a named list of (raw) lexicons, each element as a data.table or a data.frame with respectively a character column (the words) and a numeric column (the polarity scores). This argument can be one of the built-in lexicons accessible via list_lexicons.
- **valenceIn**: a single valence word list as a data.table or a data.frame with respectively a "x" and a "y" or "t" column. The first column has the words, "y" has the values for bigram shifting, and "t" has the types of the valence shifter for a clustered approach to sentiment calculation (supported types: 1 = negators, 2 = amplifiers, 3 = deamplifiers). If three columns are provided, the first two will be considered only. This argument can be one of the built-in valence word lists accessible via list_valence_shifters. A word that appears in both a lexicon and the valence word list is prioritized as a lexical entry during sentiment calculation. If NULL, valence shifting is not 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 of class sentolexicons 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 scores. The "t" column for valence shifters contains the different 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 <- sento_lexicons(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("GL_en")])
valIn <- list_valence_shifters[["en"]]
l2 <- sento_lexicons(lexIn)
l3 <- sento_lexicons(lexIn, valIn)
l4 <- sento_lexicons(lexIn, valIn[, c("x", "y")], do.split = TRUE)
l5 <- sento_lexicons(lexIn, valIn[, c("x", "t")], do.split = TRUE)
l6 <- 15[c("GI_en_POS", "valence")]

## Not run:
# include lexicons from lexicon package
lexIn2 <- list(hul = lexicon::hash_sentiment_huliu, joc = lexicon::hash_sentiment_jockers)
l7 <- sento_lexicons(c(lexIn, lexIn2), valIn)
## End(Not run)

## Not run:
# faulty extraction, no replacement allowed
15["valence"]
l2[0]
l3[22]
l4[[1]] <- l2[[1]]
l4$GI_en_NEG <- l2$myLexicon
## End(Not run)

---

sento_measures  One-way road towards a sentomeasures object

Description

Wrapper function which assembles calls to compute_sentiment and aggregate. Serves as the most direct way towards a panel of textual sentiment measures as a sentomeasures object.

Usage

sento_measures(sentocorpus, lexicons, ctr)
sentomeasures

Arguments

- `sentocorpus` a sentocorpus object created with `sento_corpus`.
- `lexicons` a sentolexicons object created with `sento_lexicons`.
- `ctr` output from a `ctr_agg` call.

Details

As a general rule, neither the names of the features, lexicons or time weighting schemes may contain any '-' symbol.

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. The "date" column has the dates converted at the frequency for across-document aggregation. All zeros are replaced by NA if `ctr$do.ignoreZeros = TRUE`.
- `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 `attributions` function. Serves further no direct purpose.

Author(s)

Samuel Borms, Keven Bluteau

See Also

- `compute_sentiment`, `aggregate`
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 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howWithin = "counts",
               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 sentiment-based 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 prediction performance metrics.

Usage

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

Arguments

sentomeasures  a sentomeasures object created using sento_measures.
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.table, data.frame or matrix 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. 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, including for example the estimated coefficients.
- `model`: the input argument `ctr$model`, to indicate the type of model estimated.
- `alpha`: calibrated \( \alpha \).
- `lambda`: calibrated \( \lambda \).
- `trained`: output from `train` call (if `ctr$type` = "cv"). There is no such output if the control parameters `alphas` and `lambdas` both specify one value.
- `ic`: a list composed of two elements: under "criterion", the type of information criterion used in the calibration, and under "matrix", a matrix of all information criterion values for `alphas` as rows and the respective `lambda` values as columns (if `ctr$type` != "cv"). Any NA value in the latter element means the specific information criterion could not be computed.
- `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`: a vector of size two, with respectively the number of sentiment measures, and the number of 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 50% 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 the out-of-sample predictions are carried out.
- `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, attributions

Examples

```r
# Not run:
data("usnews", package = "sentometrics")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")
data("epu", package = "sentometrics")
set.seed(505)

corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "2004-01-01")
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
ctr <- ctr_agg(howWithin = "counts", howDocs = "proportional",
               howTime = c("equal_weight", "linear"),
               by = "month", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

# construct a sentomeasures object to start with
corpusAll <- sento_corpus(corpusdf = usnews)
corpus <- quanteda::corpus_subset(corpusAll, date >= "2004-01-01")
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
ctr <- ctr_agg(howWithin = "counts", 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 %in% get_dates(sentomeasures), "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")

# 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 <- attributions(out1, sentomeasures,
```

to_sentiment

Convert a sentiment table to a sentiment object

Description

Converts a properly structured sentiment table into a sentiment object, that can be used for further aggregation with the aggregate.sentiment function. This allows to start from document-level sentiment scores not necessarily computed with compute_sentiment.

Usage

to_sentiment(s)
to_sentocorpus

Arguments

- \texttt{s}:
  a data.table that can be converted into a sentiment object. It should have an "id", a "date" and a "word_count" column. If other column names are provided with a separating "--", the first part is considered the lexicon (or more generally, the sentiment computation method), and the second part the feature. For sentiment column names without any "--", a "dummyFeature" component is added.

Value

A sentiment object.

Author(s)

Samuel Borms

Examples

```r
set.seed(505)

ids <- paste0("id", 1:200)
date <- sample(seq(as.Date("2015-01-01"), as.Date("2018-01-01"), by = "day"), 200, TRUE)
word_count <- sample(150:850, 200, replace = TRUE)
sent <- matrix(rnorm(200 * 8), nrow = 200)
s1 <- s2 <- s3 <- data.table(id = ids, date = date, word_count = word_count, sent)
m <- "method"

colnames(s1)[-c(1:3)] <- paste0(m, 1:8)
sent1 <- to_sentiment(s1)

colnames(s2)[-c(1:3)] <- c(paste0(m, 1:4, "--", "feat1"), paste0(m, 1:4, "--", "feat2"))
sent2 <- to_sentiment(s2)

colnames(s3)[-c(1:3)] <- c(paste0(m, 1:3, "--", "feat1"), paste0(m, 1:3, "--", "feat2"),
                          paste0(m, 4:5))
sent3 <- to_sentiment(s3)

# further aggregation from then on is easy
sentMeas1 <- aggregate(sent1, ctr_agg(lag = 10))
```

Description

Convert a quanteda corpus object into a sentocorpus object.
Usage

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

Arguments

corpus a quanteda corpus object.
dates a sequence of dates as "yyy-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
weights_almon

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

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

weights_almon

Description
Computes Almon polynomial weighting curves. Handy to self-select specific time aggregation
weighting schemes for input in ctr_agg using the weights argument.

Usage
weights_almon(n, orders = 1:3, do.inverse = TRUE,
do.normalize = TRUE)
weights_beta

Arguments

- **n**: a single numeric to indicate the lag length (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: \((1 - (1 - i/n)^b)(1 - i/n)^{B-b}\), where \(i\) is the lag index ordered from 1 to \(n\). 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`

Description

Computes Beta weighting curves as in Ghysels, Sinko and Valkanov (2007). Handy to self-select specific time aggregation weighting schemes for input in `ctr_agg` using the weights argument.

Usage

```r
weights_beta(n, a = 1:4, b = 1:4)
```

Arguments

- **n**: a single numeric to indicate the lag length (cf., n).
- **a**: a numeric as the first parameter (cf., a).
- **b**: a numeric as the second parameter (cf., b).

Details

The Beta weighting abides by following formula: \(f(i/n; a, b)/\sum_i f(i/n; a, b)\), where \(i\) is the lag index ordered from 1 to \(n\), \(a\) and \(b\) are two decay parameters, and \(f(x; a, b) = (x^{a-1}(1-x)^{b-1}\Gamma(a+b))/(\Gamma(a)\Gamma(b))\), where \(\Gamma(.)\) is the gamma function.
weights_exponential

Value

A data.frame of beta weighting curves per combination of a and b. If n = 1, all weights are set to 1.

References


See Also

ctr_agg

weights_exponential  Compute exponential weighting curves

Description

Computes exponential weighting curves. Handy to self-select specific time aggregation weighting schemes for input in ctr_agg using the weights argument.

Usage

weights_exponential(n, alphas = seq(0.1, 0.5, by = 0.1))

Arguments

n  a single numeric to indicate the lag length.
alphas  a numeric vector of decay factors.

Value

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

See Also

ctr_agg
Index

*Topic datasets
  epu, 16
  list_lexicons, 21
  list_valence_shifters, 22
  usnews, 49

add_features, 3, 4, 40
aggregate, 42, 43
aggregate.sentiment, 3, 6, 9, 47
almons, 12
attributions, 3, 7, 33, 43, 46
compute_sentiment, 3, 6, 8, 11, 12, 32, 42, 47

corpus, 4, 8, 9, 39, 40, 48, 49
corpus_sample, 40
corpus_subset, 40
ctr_agg, 3, 6, 10, 18, 43, 50–52
ctr_model, 3, 12, 44, 46
diff, 15
diff.sentomeasures, 15
docvars, 8
epu, 16

gamma, 51
generate_dates, 17
generate_dimensions, 17
generate_hows, 8, 11, 12, 18
generate_loss_data, 18
get.measures, 20
ggplot, 33–35
glmnet, 14, 45, 46

hash_valence_shifters, 22

list_lexicons, 21
list_valence_shifters, 22

MCSprocedure, 19

measures_delete, 23, 29
measures.fill, 11, 12, 24
measures_global, 25
measures_merge, 26
measures_select, 23, 28
measures_subset, 29

nmeasures, 30
peakdates, 3, 31
peaksdocs, 3, 32
plot.attributions, 33
plot.sentomeasures, 34
plot.sentomodeliter, 35
predict.glmnet, 36
predict.sentomodel, 3, 36

scale, 37
scale.sentomeasures, 37

sentiment_bind, 38

tenko_corpus, 3, 4, 8, 39, 43, 49, 50
tenko_lexicons, 3, 8, 9, 41, 43
tenko_measures, 3, 6, 7, 15, 17, 20, 23–25,
  27–31, 34, 37, 42, 44
tenko_model, 3, 7, 8, 13, 14, 19, 26, 33, 35,
  36, 44
tentometrics (sentometrics-package), 3
tentometrics-package, 3
setThreadOptions, 9
stopwords, 21, 22

to.sentiment, 6, 32, 47
to.sentocorpus, 48
tokenize_words, 9
tokens, 8

train, 13, 45, 46

usnews, 49

weights_almon, 11, 50
weights_beta, 11, 51
weights_exponential, 11, 52