Package 'sentometrics'

October 17, 2017

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

Version 0.2.0

Description Time series analysis based on textual sentiment, accounting for the intrinsic challenge that sentiment can be computed and pooled across texts and time in many ways. Provides a means to model the impact of sentiment in texts on a target variable, by first computing a wide range of textual sentiment measures and then selecting those that are most informative.

Depends R (>= 3.4.0), data.table, ggplot2

License GPL-2

Encoding UTF-8

LazyData true

Suggests testthat, e1071, randomForest

Imports utils, stats, quanteda, sentimentr, stringi, zoo, abind, glmnet, caret, compiler, Rcpp, RcppRoll, ggthemes, ISOweek, MCS

RoxygenNote 6.0.1

R topics documented:

`add_features` *Add feature columns to a sentocorpus*

Description

Adds new named feature columns to provided `sentocorpus` object.

Usage

```
add_features(sentocorpus, featuresdf)
```

Arguments

`sentocorpus`	a `sentocorpus` object.
`featuresdf`	a named `data.table` (or `data.frame`) with as columns the new features of type `numeric` to add to the `sentocorpus` inputted. If the number of rows in `featuresdf` is not equal to the number of documents in `sentocorpus`, recycling will occur.

Value

An updated `sentocorpus` object.

Author(s)

Samuel Borms

Examples

```
data("useconomynews")

# construct a corpus and add random features to it
corpus <- sento_corpus(corpusdf = useconomynews)
corpus <- add_features(corpus,
                       featuresdf = data.table(random = runif(quanteda::ndoc(corpus))))
```

almons *Compute Almon polynomials*

Description

Computes Almon polynomial weighting curves; handy to self-select specific time aggregation weighting schemes.

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* in the formula).
orders	a numeric vector as the sequence the Almon orders (cf. *b* in the formula).
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: $(1 - (i/n)^b)(i/n)^{max(b)-b}$.

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 scheme. Relies partly on the **quanteda** package. The scores computed are net sentiment (sum of positive minus sum of negative scores).

Usage

```
compute_sentiment(sentocorpus, lexicons, how = get_hows()$words, dfm = NULL)
```

Arguments

sentocorpus	a sentocorpus object.
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 occer, see get_hows()$words.
dfm	optional; an output from a quanteda::dfm() call, such that users can specify their own tokenization scheme (via quanteda::tokenize()) as well as other parameters related to the construction of a document-feature matrix (dfm). By default, a dfm is created based on a tokenization that removes punctuation, numbers, symbols and separators. We suggest to stick to unigrams, as the remainder of the sentiment computation and built-in lexicons assume the same.

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.

Value

A list containing:

corpus	the supplied sentocorpus object.
sentiment	the sentiment scores data.table with a date and lexicon–feature sentiment scores columns.
features	a character vector of the different features.
lexicons	a character vector of the different lexicons used.
howWithin	a character vector to remind how sentiment within documents was aggregated.

Author(s)

Samuel Borms

See Also

dfm, tokenize

Examples

```
data("useconomynews")
data("lexicons")
data("valence")

# sentiment computation based on raw frequency counts
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
sent <- compute_sentiment(corpus, l, how = "counts")

# same sentiment computation based on a user-supplied dfm with default settings
dfm <- quanteda::dfm(quanteda::tokenize(corpus), verbose = FALSE)
```

```
sent <- compute_sentiment(corpus, l, how = "counts", dfm = dfm)
```

`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 = FALSE, by = "day", lag = 1,
  fill = "zero", alphasExp = seq(0.1, 0.5, by = 0.1), ordersAlm = 1:3,
  do.inverseAlm = TRUE, do.normalizeAlm = TRUE, weights = NULL,
  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 occer, see `get_hows()$words`.
`howDocs`	a single `character` vector defining how aggregation across documents per date will be performed. Should `length(howDocs) > 1`, the first element is used. For currently available options on how aggregation can occer, see `get_hows()$docs`.
`howTime`	a `character` vector defining how aggregation across dates will be performed. More than one choice is possible here. For currently available options on how aggregation can occer, 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`	a single `character` vector, either `"day"`, `"week"`, `"month"` or `"year"`, to indicate at what level the dates should be aggregated. Dates will be 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.
`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 `fill_measures()` 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 betwoon 0 and 1 (both excluded).
`ordersAlm`	a `numeric` vector of all Almon polynomial orders to calcalute weights for, used if `"almon" %in% howTime`.

do.inverseAlm — a logical indicating if for every Almon polynomial its inverse has to be calculated too, used if "almon" %in% howTime.

do.normalizeAlm — a logical indicating if every Almon polynomial weights column should sum to one, 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 and exponential weighting functions are created sequentially; if the user wants only specific of such time weighting series it can use almons() and exponentials(), select the columns it requires, combine it into a data.frame and supply it under this argument (see examples).

dfm — optional; an output from a quanteda::dfm() call, such that users can specify their own tokenization scheme (via quanteda::tokenize()) as well as other parameters related to the construction of a document-feature matrix (dfm). By default, a dfm is created based on a tokenization that removes punctuation, numbers, symbols and separators. We suggest to stick to unigrams, as the remainder of the sentiment computation (in compute_sentiment()) and built-in lexicons assume the same.

Details

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

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

fill_measures, 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 = "tf-idf",
                howDocs = "proportional",
                howTime = c("equal_weight", "linear", "almon", "exponential", "own"),
                do.ignoreZeros = TRUE,
                by = "day",
                lag = 20,
                ordersAlm = 1:3,
                do.inverseAlm = TRUE,
                do.normalizeAlm = TRUE,
                alphasExp = c(0.20, 0.50, 0.70, 0.95),
                weights = data.frame(myWeights = runif(20)))
```

```
# set up control function with one linear, two Almon and two exponential weighting schemes
a <- almons(n = 70, orders = 1:3, do.inverse = TRUE, do.normalize = TRUE)
e <- exponentials(n = 70, alphas = c(0.4, 0.8))
ctr3 <- ctr_agg(howTime = c("linear"), by = "year",
                weights = data.frame(a1 = a[, 1], a2 = a[, 3], e1 = e[, 1], e2 = e[, 2]))
```

`ctr_merge` *Set up control for merging sentiment measures*

Description

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

Usage

```
ctr_merge(sentomeasures, feat = NA, lex = NA, time = NA,
  do.keep = FALSE)
```

Arguments

`sentomeasures`	a `sentomeasures` object. This is necessary to check whether the other input arguments make sense.
`feat`	a list with unique features to merge at given name, e.g. `list(feat12 = c("feat1", "feat2"))`. See `sentomeasures$features` for the exact names to use.
`lex`	a list with unique lexicons to merge at given name, e.g. `list(lex12 = c("lex1", "lex2"))`. See `sentomeasures$lexicons` for the exact names to use.
`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.
`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 `TRUE`).

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms

See Also

`merge_measures`

Examples

```
data("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

# set up a correct control function
ctrMerge <- ctr_merge(sentomeasures,
                      time = list(W = c("equal_weight", "linear")),
                      lex = list(LEX = c("LM_eng", "HENRY_eng")),
                      feat = 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")),
                      lex = list(LEX = c("LM_eng", "HENRY_eng")),
                      feat = 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 the `sento_model` function's documentation for details on the estimation and calibration procedure.

Usage

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

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 selection criteria to use. Currently 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.

`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 forecasting exercise through time.

`h` an `integer` value to shift the time series to have the desired (forecasting) 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.

`alphas` a `numeric` vector of the different alphas to test for during optimization, between 0 and 1. A value of 0 pertains to Ridge optimization, a value of 1 to LASSO optimization; values in between are pure elastic net. The lambda values tested for are automatically 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 calibration at every iteration (ignored if `iter == FALSE`).

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

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

`oos` a non-negative `integer` to indicate the number of periods to skip from the end of the cross-validation training sample (out-of-sample) up to the test sample (ignored if `type != "cv"`).

`start` a positive `integer` to indicate at which point the iteration has to start (ignored if `iter == FALSE`). For example, for 100 out-of-sample iterations, `start = 70` only performs the analysis for the last 31 samples.

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

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

`sento_model`

Examples

```
# series of example 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 = "BIC", do.iter = TRUE, h = 0, nSample = 100)

# series of example 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 = 0, trainWindow = 250,
                    testWindow = 4, oos = 0, 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)
```

---

epu *Monthly Economic Policy Uncertainty Index*

Description

Monthly values of a news-based index of US 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 in a mutually exclusive sense.

Usage

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

Format

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

Source

Research on Economic Policy Uncertainty

`exponentials` *Compute exponential weighting curves*

Description

Computes exponential weighting curves; handy to self-select specific time aggregation weighting schemes.

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

---

fill_measures *Add and fill missing dates*

Description

Adds missing dates between earliest and latest date, such that time series is continuous on a period-by-period basis. Fills in these dates with either 0, the respective latest non-missing value or NA.

Usage

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

Arguments

sentomeasures	a sentomeasures object.
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("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

# fill measures
```

```
f1 <- fill_measures(sentomeasures)
f2 <- fill_measures(sentomeasures, fill = "latest")
f3 <- fill_measures(sentomeasures, fill = NA)
```

`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()
```

Value

A list with the supported aggregation hows for arguments `howWithin` (within documents), `howDows` (across documents, per date) and `howTime` (across dates), to be supplied to `ctr_agg()`.

See Also

`ctr_agg`

`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, and `"_tr"` as suffix if the lexicon is translated. The lexicons are in the form required for further sentiment analysis. The built-in lexicons are the following:

- FEEL_eng_tr (FEEL: French Expanded Emotion Lexicon)
- FEEL_fr
- FEEL_nl_tr
- GI_eng (GI: General Inquirer, i.e. Harvard IV-4 combined with Laswell)
- GI_fr_tr
- GI_nl_tr
- HENRY_eng (HENRY: Henry)
- HENRY_fr_tr
- HENRY_nl_tr
- LM_eng (LM: Loughran and McDonald)
- LM_fr_tr
- LM_nl_tr

Usage

```
data("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

```
lexicons[c("FEEL_eng_tr", "LM_eng")]
```

---

merge_measures *Merge sentiment measures*

Description

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

Usage

```
merge_measures(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("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

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

`perform_agg` *Aggregate textual sentiment across documents and time*

Description

Condense document-level textual sentiment scores into a panel of textual sentiment measures by aggregating across documents and time.

Usage

```
perform_agg(toAgg, ctr)
```

Arguments

`toAgg`	output from a `compute_sentiment()` call, a list with as main component a sentiment scores `data.table` with dates and lexicon–feature sentiment scores columns.
`ctr`	output from a `ctr_agg()` call.

Value

A `sentomeasures` object.

Author(s)

Samuel Borms, Keven Bluteau

See Also

`compute_sentiment`, `ctr_agg`

Examples

```
data("useconomynews")
data("lexicons")
data("valence")

# computation of sentiment and aggregation into sentiment measures
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
sent <- compute_sentiment(corpus, l, how = "counts")
ctr <- ctr_agg(howTime = c("linear"), by = "year", lag = 3)
sentomeasures <- perform_agg(sent, ctr)
```

`perform_MCS` *Apply the 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"` or `"multinomial"` models.
`...`	other parameters that can be supplied to the `MCS::MCSprocedure()` function.

Value

An object as returned from the `MCS::MCSprocedure()` function.

Author(s)

Samuel Borms

See Also

`sento_model`, `MCSprocedure`

`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 codeselect_measures() 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.
`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 **ggplot2** plot, 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("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

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

# adjust appearance of plot
p <- plot(sentomeasures)
```

```
p <- p +
  ggthemes::theme_base() +
  scale_x_date(name = "date") +
  scale_y_continuous(name = "newName")
p
```

`plot.sentomodeliter` *Plot iterative forecasts versus realized values*

Description

Displays a plot of all forecasts 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 `sentomeasures` object.
`...`	not used.

Value

Returns a simple **ggplot2** plot, which can be added onto (or to alter its default elements) by using the + operator (see examples).

Author(s)

Samuel Borms

Examples

```
data("useconomynews")
data("lexicons")
data("valence")
data("epu")

# construct a sentomeasures object to start with
useconomynews <- useconomynews[date >= "1980-01-01" & date < "2014-10-01", ]
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howWithin = "tf-idf", howDocs = "proportional",
               howTime = c("equal_weight", "linear", "almon"),
               by = "month", lag = 3, ordersAlm = 1:3,
               do.inverseAlm = TRUE, do.normalizeAlm = TRUE)
sentomeasures <- sento_measures(corpus, l, ctr)

# prepare y and other x variables
y <- epu[epu$date >= sentomeasures$measures$date[1], ]$index
```

```
length(y) == nrow(sentomeasures$measures) # TRUE
x <- data.frame(runif(length(y)), rnorm(length(y))) # two other (random) x variables
colnames(x) <- c("x1", "x2")

# estimate regression iteratively based on a sample of 120, skipping the first 275 samples
ctr <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                 h = 0, nSample = 120, start = 276)
out <- sento_model(sentomeasures, y, x = x, ctr = ctr)
summary(out)

# plotting
p <- plot(out)
p <- p +
  ggthemes::theme_few()
p
```

`plot_attributions` *Plot forecasting attribution at specified level*

Description

Shows a plot of the attributions along the dimension provided.

Usage

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

Arguments

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

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 de facto they appear only once at one date (even though a document may contribute to forecasts at several dates, depending on the number of lags in the time aggregation.)

Value

Returns a simple **ggplot2** plot, 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 dimensions (thus, individual plots) is at maximum twelve.

Author(s)

Samuel Borms

`predict.sentomodel` *Make predictions from a sentomodel object*

Description

Prediction (forecasting) 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.
`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 provided.

Author(s)

Samuel Borms

See Also

`predict.glmnet`, `sento_model`

`retrieve_attributions` *Retrieve top-down sentiment attributions given forecasting model object*

Description

Computes the attributions to forecasts 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, ...)
```

Arguments

`model`	a `sentomodel` or `sentomodeliter` object.
`sentomeasures`	the `sentomeasures` object used to estimate the `sentomodel` object argument.
`...`	the dates at which attribution is to be performed. The dates should be between the latest date avaiable in the input `sentomeasures` object and the first date of the sample used to estimate the model in the `model` argument. If not provided, attribution will be calculated for all in-sample dates. Not used if `model` is a `sentomodeliter` object, for which attribution for all out-of-sample forecasting dates is calculated.

Details

See `sento_model` for an elaborate modelling example including the calculation and plotting of attributions.

Value

A list with all possible dimensions for which aggregation is computed, being `"documents"`, `"lexicons"`, `"features"`, `"time"` and `"lags"`. The last four dimensions are `data.tables` having a `"date"` column and the other columns the different names of the dimension, with the attributions as values. For document-level attribution, the list 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. `NA`s are removed first.

Usage

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

Arguments

`x`	a `sentomeasures` object.
`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("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

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

`select_measures` *Select a subset of sentiment measures*

Description

Selects the subset of 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. Selecting a subset of dates

Usage

```
select_measures(sentomeasures, toSelect = "all", do.combine = TRUE,
  dates = NA)
```

Arguments

`sentomeasures`	a `sentomeasures` object.
`toSelect`	a `"character"` vector of the lexicon, feature and time weighting scheme names, to indicate which measures need to be selected. By default equal to `"all"`, which means no selection of the sentiment measures is made; this may be used if one only wants to extract a subset of dates via the `dates` argument.
`do.combine`	a `logical` indicating if only measures for wich all (`TRUE`) or at least one (`FALSE`) of the selection components should occur in each sentiment measure's name in the subset. If `do.combine == TRUE`, the `toSelect` argument can only consist of one lexicon, one feature and one time weighting scheme at maximum.
`dates`	any expression, in the form of a `character` vector, that would correctly evaluate to a `logical` vector, features the variable `date` and has dates specified as `"yyyy-mm-dd"`, e.g. `dates = "date >= '2000-01-15'"`. This argument may also be a vector of class `Date` which extracts all dates that show up in that vector. See the examples. By default equal to `NA`, meaning no subsetting based on dates is done.

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("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
```

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

# different selections
sel1 <- select_measures(sentomeasures, c("equal_weight"))
sel2 <- select_measures(sentomeasures, c("equal_weight", "linear"), do.combine = FALSE)
sel3 <- select_measures(sentomeasures, c("linear", "LM_eng"))
sel4 <- select_measures(sentomeasures, c("linear", "LM_eng", "wsj", "economy"),
                        do.combine = FALSE)
sel5 <- select_measures(sentomeasures, c("linear", "LM_eng"),
                        dates = "date >= '1989-12-31' & date <= '2000-12-31'")
d <- seq(as.Date("1980-01-01"), as.Date("2013-12-01"), by = "month")
sel6 <- select_measures(sentomeasures, c("linear", "LM_eng"), dates = d)
```

`sentometrics` *An Integrated Framework for Textual Sentiment Time Series Aggregation and Forecasting*

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 hundreds of different 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 packages. As a final outcome, this package 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 the sentiment times series that are most informative. The package created therefore integrates the qualification of sentiment from texts, the aggregation into different sentiment measures and the optimized forecasting based on these measures.

Main functions

- Sentiment computation and aggregation into sentiment measures: `sento_corpus`, `ctr_agg`, `compute_sentiment`, `sento_measures`, `to_global`
- Sparse modelling: `ctr_model`, `sento_model`
- Forecasting and post-modelling analysis: `retrieve_attributions`, `perform_MCS`

Update

The latest version of the package is available at https://github.com/ArdiaD/Sentometrics.

Note

The ideas behind the sentiment aggregation framework can be consulted in the working paper titled "Questioning the news about economic growth: Sparse forecasting using thousands of news-based sentiment values" (Ardia, Bluteau & Boudt, 2017) at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2976084.

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

Author(s)

Samuel Borms, Keven Bluteau, David Ardia and Kris Boudt.

sento_corpus *Create a sentocorpus object*

Description

Formalizes the structure of a collection of texts into a well-defined corpus, by calling the corpus() constructor from the **quanteda** package, performing a set of checks and preparing it for further sentiment analysis.

Usage

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

Arguments

corpusdf	a data.table (or data.frame) with as named columns and *in this order*: a document id column, a date column, a text 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 are often binary (1 means the feature is applicable to the document in the same row) or as a percentage 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 no particular features are of interest to your analysis, have only one additional column with all values set to 1. Provide the date column as "yyyy-mm-dd". All spaces in the names of the features are automatically replaced by underscores.
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.

Details

A sentocorpus object can be regarded as 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 quanteda::corpus_sample() or quanteda::corpus_subset(), that do not change the actual corpus structure and may come in handy. To add additional additional features, we recommend to use add_features(). In the future, we will formalize the interaction between the **quanteda** package (as well as other text mining packages).

Value

A sentocorpus object, derived from a **quanteda** corpus classed list keeping the elements documents, metadata andsettings. The documents element incorporates the corpus represented as a data.table.

Author(s)

Samuel Borms

See Also

corpus

Examples

```
data("useconomynews")

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

# take a random subset using a quanteda's package function
corpusSmall <- quanteda::corpus_sample(corpus, size = 500)
```

---

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, and a `sentomeasures` object.

Usage

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

Arguments

`sentocorpus`	a `sentocorpus` object.
`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 a `date` and lexicon–feature sentiment scores columns.
`howWithin`	a `character` vector to remind how sentiment within documents was aggregated.

howDocs	a 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.
attribWeights	a list of document weights and time weights that are extracted in a call to retrieve_attributions(). Serves further no direct purpose.

Author(s)

Samuel Borms, Keven Bluteau

See Also

compute_sentiment, perform_agg

Examples

```
data("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howWithin = "tf-idf",
               howDocs = "proportional",
               howTime = c("equal_weight", "linear", "almon"),
               by = "month",
               lag = 3,
               ordersAlm = 1:3,
               do.inverseAlm = TRUE,
               do.normalizeAlm = TRUE)
sentomeasures <- sento_measures(corpus, 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 forecasting performance metrics.

Usage

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

Arguments

sentomeasures	a sentomeasures object. 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 logistic 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). In the latter case, the training metric is automatically set to "RMSE" for a linear model and to "Accuracy" for a logistic model. We suppressed many of the details that can be supplied to the glmnet::glmnet() and caret::train() functions we rely on for estimation and calibration through cross-validation, 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$model, to remind of the type of model that was estimated.
x	the matrix of the values used in the regression for all explanatory variables.
alpha	optimized calibrated alpha.
lambda	optimized calibrated lambda.
trained	output from caret::train call (if ctr$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").
date	a reference date, being the 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.

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 found at sentomodel$date.
alphas	optimized calibrated alphas.
lambdas	optimized 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 forecast between two consecutive time points (omitting the very first forecast, 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

Examples

```
data("useconomynews")
data("lexicons")
data("valence")
data("epu")

# construct a sentomeasures object to start with
useconomynews <- useconomynews[date >= "1980-01-01" & date < "2014-10-01", ]
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howWithin = "tf-idf", howDocs = "proportional",
               howTime = c("equal_weight", "linear", "almon"),
               by = "month", lag = 3, ordersAlm = 1:3,
               do.inverseAlm = TRUE, do.normalizeAlm = TRUE)
sentomeasures <- sento_measures(corpus, l, ctr)

# prepare y and other x variables
y <- epu[epu$date >= sentomeasures$measures$date[1], ]$index
length(y) == nrow(sentomeasures$measures) # TRUE
x <- data.frame(runif(length(y)), rnorm(length(y))) # two other (random) x variables
colnames(x) <- c("x1", "x2")
# a list with models based on the three implemented information criteria
out1 <- list()
for (ic in c("BIC", "AIC", "Cp")) {
 ctrIC <- ctr_model(model = "gaussian", type = ic, do.iter = FALSE, h = 0)
 out1[[ic]] <- sento_model(sentomeasures, y, x = x, ctr = ctrIC)
}
```

```
# a cross-validation based model
ctrCV <- ctr_model(model = "gaussian", type = "cv", do.iter = FALSE,
                   h = 0, alphas = c(0.10, 0.50, 0.90), trainWindow = 350,
                   testWindow = 40, oos = 0, do.progress = TRUE)
out2 <- sento_model(sentomeasures, y, x = x, ctr = ctrCV)
summary(out2)

# a cross-validation based model but 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 = 350,
                    testWindow = 40, oos = 0, do.progress = TRUE)
out3 <- sento_model(sentomeasures, yb, x = x, ctr = ctrCVb)
summary(out3)

# an example of an iterative analysis
ctrIter <- ctr_model(model = "gaussian", type = "BIC", do.iter = TRUE,
                       h = 0, nSample = 300, start = 106)
out <- sento_model(sentomeasures, y, x = x, ctr = ctrIter)
summary(out)

# some post-analysis (attribution and prediction)
attributions <- retrieve_attributions(out, sentomeasures)
plot_attributions(attributions, "lexicons")
plot_attributions(attributions, "features")
plot_attributions(attributions, "time")

nx <- ncol(sentomeasures$measures) - 1 + ncol(x) # don't count date column
newx <- runif(nx) * cbind(sentomeasures$measures[, -1], x)[1:nrow(x), ]
preds2 <- predict(out2, newx = as.matrix(newx), type = "link")
preds3 <- predict(out3, newx = as.matrix(newx), type = "class")
```

`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("lexicons")` or a valence word list from `data("valence")` as an argument. Makes use of the `as_key()` function from the **sentimentr** package to make the output coherent and check for duplicates.

Usage

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

Arguments

`lexiconsIn`	a list of (raw) lexicons, each element being a `data.table` or `data.frame` with respectively a words column and a polarity score column. The lexicons should be appropriately named for clarity in terms of subsequently obtained sentiment measures. Alternatively, a subset of the already formatted built-in lexicons accessible via `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, ony can simply supply `lexicons[c(...)]` as an argument to either `sento_measures()` or `compute_sentiment()`. However, it is strongly recommended to pass the lexicons (and a valence word list) that want to be used through this function.

`valenceIn` a single valence word list as a `data.table` or `data.frame` with respectively a words column, a type column (1 for negators, 2 for amplifiers/intensifiers, and 3 for deamplifiers/downtoners) and a score column. Suggested scores are -1, 2 and 0.5 respectively, and should be the same within each type. Alternatively, this argument can be one of the already formatted built-in valence word lists accessible via `valence`. If `NULL`, no valence word list is part of the output.

`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 a list 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. Lexicon scores are multiplied by -1, 2 and 0.5 by default, respectively, or the first value of the scores column of the valence word list.

Author(s)

Samuel Borms

See Also

`as_key`

Examples

```
data("lexicons")
data("valence")

# sets up output list straight from built-in word lists including valence words
l1 <- c(lexicons[c("LM_eng", "HENRY_eng")], valence[["eng"]])

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

sp500 *Monthly S&P 500 Index returns*

Description

Monthly returns for the S&P 500 Index between March 1988 and December 2014, including a binomial and a multinomial example series. It has following columns:

- date. Date as `"yyyy-mm-01"`.
- return. A `numeric` value as the return that was achieved during the corresponding month; for example the first return value is the price change from beginning of March 1988 to beginning of April 1988.
- up. A `factor` with value `"pos"` if the return is greater than zero, else `"neg"`.
- upMulti. A `factor` with values `"pos+"`, `"pos"`, `"neg"` and `"neg-"` if returns are greater than 0.05 and 0, or smaller than 0 and -0.05, respectively and in a mutually exclusive sense.

Usage

```
data("sp500")
```

Format

A `data.frame` with 322 rows and 4 columns.

Source

S&P 500 (^GSPC) at Yahoo Finance

to_global *Merge sentiment measures into one global sentiment measure*

Description

Merges all sentiment measures into one global textual sentiment measure based on a set of weights to indicate the importance of each component in the `lexicons`, `features` and `time` vectors as specified in the input `sentomeasures` object. The global measure is composed as the multiplication of the individual weights across the three dimensions times the sentiment value per date observation.

Usage

```
to_global(sentomeasures, lex = 1, feat = 1, time = 1)
```

Arguments

sentomeasures	a sentomeasures object.
lex	a numeric vector of weights, of size length(sentomeasures$lexicons), in the same order and summing to one. By default set to 1, which means equally weighted.
feat	a numeric vector of weights, of size length(sentomeasures$features), in the same order and summing to one. By default set to 1, which means equally weighted.
time	a numeric vector of weights, of size length(sentomeasures$time), in the same order and summing to one. By default set to 1, which means equally weighted.

Details

This function returns no sentomeasures object, however the global sentiment measure as outputted can be added to regressions as an additional variable using the x argument in the sento_model() function.

Value

A data.frame with the values for the global sentiment measure under the global column and dates as row names.

Author(s)

Samuel Borms

See Also

sento_model

Examples

```
data("useconomynews")
data("lexicons")
data("valence")

# construct a sentomeasures object to start with
corpus <- sento_corpus(corpusdf = useconomynews)
l <- setup_lexicons(lexicons[c("LM_eng", "HENRY_eng")], valence[["valence_eng"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sentomeasures <- sento_measures(corpus, l, ctr)

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

useconomynews *Texts relevant (and not) to the US economy*

Description

A collection of texts annotated by humans in terms of relevance to the US economoy or not. The texts come from two major journals in the US (The Wall Street Journal and The Washington Post) and cover 6801 documents between 1980 and 2014. It contains following information:

- id. ID identifier.
- date. Date as `"yyyy-mm-dd"`.
- text. 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.
- economy. Equals 1 if the article is relevant to the US economy.
- noneconomy. Equals 1 if the article is not relevant to the US economy.

Usage

```
data("useconomynews")
```

Format

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

Source

Economic News Article Tone and Relevance

valence *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 value associated to a word and type of valence shifter. The list element names incorporate the language of the valence word list. All non-English word lists are translated. The valence word lists are in the form required for further sentiment analysis. The built-in valence word lists are the following:

- valence_eng
- valence_fr
- valence_nl

Usage

```
data("valence")
```

Format

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

Source

`hash_valence_shifters` (negators)

Index