Package 'sentometrics'

August 28, 2017

Title An Integrated Framework for Textual Sentiment Based Multivariate Time Series Modeling and Forecasting

Version 0.1.0

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

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.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.

Examples

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

`compute_sentiment` *Computation of 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)
```

Arguments

`sentocorpus`	a `sentocorpus` object.
`lexicons`	output from a `setup_lexicons()` call.
`how`	a single `character` vector defining how aggregation within documents will be performed. For currently available options on how aggregation can occer, see `get_hows()$words`.

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 automatically splits any lexicon into positive and negative polarity. `NA`s are converted to 0, under the assumption that this is equivalent to no sentiment.

Value

A list containing:

`corpus`	the supplied `sentocorpus` object.
`sentiment`	a sentiment scores `data.table` with a `date` and feature–lexicon 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.

Examples

```
# sentiment computation based on raw frequency counts
data("useconomynews")
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_agg` *Setup 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 = "equal_weight", howDocs = "equal_weight",
  howTime = "equal_weight", do.ignoreZeros = FALSE, by = "day", lag = 1,
  alphasExp = seq(0.1, 0.5, by = 0.1), ordersAlm = 1:3,
  do.inverseAlm = TRUE, do.normalizeAlm = TRUE, weights = NULL)
```

Arguments

howWithin	a single character vector defining how aggregation within documents will be performed. Should length(howWithin) > 1, the first element is used.
howDocs	a single character vector defining how aggregation across documents per date will be performed. Should length(howDocs) > 1, the first element is used.
howTime	a character vector defining how aggregation across dates will be performed. More than one choice is possible here.
do.ignoreZeros	a logical indicating whether zero sentiment values have to be ignored while aggregation 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.
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 own weighting scheme as a data.frame with the number of rows equal to the desired lag, used if "own" %in% howTime.

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.

See Also

get_hows

Examples

```
# simple control function (using equal_weight default options)
ctr1 <- ctr_agg(howTime = c("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)))
```

`ctr_merge` *Setup 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.
`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.

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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 *Setup control for sentiment measures-based regression modelling*

Description

Sets up control object for linear or nonlinear modelling of a response variable onto a sparse panel of textual sentiment measures (and potentially other variables). Models are computed using the elastic-net regularization as implemented in the **glmnet** package, to account for the sparsity of the sentiment measures. 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).

Usage

```
ctr_model(model = c("lm", "binomial", "multinomial"), type = c("BIC", "AIC",
  "Cp", "cv"), do.iter = FALSE, h = 1, lambdas = 10^seq(2, -2, length.out
  = 50), 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: "`lm`" (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. Zou, Hastie, Tibshirani et al. (2007). "On the 'degrees of freedom' of the LASSO."), and "`cv`" (cross-validation based on the `train` function from the **caret** package). The adapted information criteria are currently only available for a linear regression.
`do.iter`	a `logical`, `TRUE` induces an iterative optimization of models 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.
`lambdas`	a `numeric` vector of the different lambdas to test for during optimization.

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

Value

A list encapsulating the control parameters.

Examples

```
# series of example information criterion based model control functions
ctrIC1 <- ctr_model(model = "lm", type = "BIC", do.iter = FALSE, h = 0,
                    alphas = seq(0, 1, by = 0.10))
ctrIC2 <- ctr_model(model = "lm", type = "BIC", do.iter = TRUE, h = 0, nSample = 100)

# series of example cross-validation based model control functions
ctrCV1 <- ctr_model(model = "lm", 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 = "lm", type = "cv", do.iter = TRUE, h = 0, trainWindow = 45,
                    testWindow = 4, oos = 0, nSample = 70, do.progress = TRUE)
```

`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` or the respective latest non-missing value.

Usage

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

Arguments

sentomeasures	a sentomeasures object.
fill	an element of c("zero", "latest"), the first assumes missing dates represent zero sentiment, the latter assumes missing dates represent constant sentiment.

Value

A modified sentomeasures object.

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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")
```

---

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.

See Also

`ctr_merge`

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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)
```

`merge_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 provided to indicate the importance of each component in the `lexicons`, `features` and `time` vectors as part of the `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

```
merge_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.

Value

A modified `sentomeasures` object, with only the global sentiment measure, including updated statistics, but the other list elements and the original sentiment scores `data.table` untouched.

See Also

`ctr_merge`

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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 <- merge_to_global(sentomeasures, lex = c(0.40, 0.60),
                                         feat = c(0.10, 0.20, 0.30, 0.40),
                                         time = 1)
```

`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 feature–lexicon sentiment scores columns.
`ctr`	output from a `ctr_agg()` call.

Value

A `sentomeasures` object.

See Also

`compute_sentiment`, `ctr_agg`

Examples

```
# computation of sentiment and aggregation into sentiment measures
data("useconomynews")
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)
```

`plot.sentomeasures` *Plot sentiment measures*

Description

Straightforward plotting method for all sentiment measures in the provided `sentomeasures` object, shown in one plot. We suggest to make use of the `select_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("lexicon", "feature", "time", "all")`. The first three choices display all measures from the same group in the same color. The choice `"all"` displays every single sentiment measure in a separate color, but this may look visually overwhelming very fast.
`...`	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).

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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 = "lexicon")

# adjust appearance of plot
p <- plot(sentomeasures)
p <- p + theme_dark() +
  scale_x_date(name = "date") +
  scale_y_continuous(name = "newName")
p
```

---

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 for all explanatory variables at which predictions are to be made, see documentation for `predict.glmnet`.
`type`	type of prediction required, a value from `c("link", "response", "class")`, see documentation for `predict.glmnet`.
`offset`	values to use as offset, only if an offset was also used in the model fitting, see documentation for `predict.glmnet`.
`...`	not used.

Value

A prediction output depending on the `type` argument provided.

See Also

`predict.glmnet`, `sento_model`

`retrieve_attributions` *Retrieves top-down sentiment attribution given forecasting model*

Description

### TODO

Usage

```
retrieve_attributions(sentomodel)
```

Arguments

`sentomodel`	a `sentomodel` object.

Value

A list with all possible dimensions for which aggregation can be computed, as `data.tables` with an `"id"` (at document-level) column, a `"date"` column and a `"attribution"` column. ### TODO

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.

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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.

Usage

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

Arguments

`sentomeasures`	a `sentomeasures` object.
`toSelect`	a vector of components (lexicon, time weighting and feature names) which form the measures selected.
`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.

Value

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

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
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)
```

`sentometrics` *An Integrated Framework for Textual Sentiment Based Multivariate Time Series Modeling 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.

Functions

- Sentiment computation and aggregation into sentiment measures: to do
- Sparse modelling: to do
- Forecasting and post-modelling analysis: to do

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

Assembles a collection of texts (i.e. a corpus), by calling the `corpus()` instructor from the **quanteda** package, perform a set of checks and prepare it for further 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 for example topics (e.g. economic, political or legal), but also article sources (e.g. online or printed press), amongst many more possibilities. Feature column names should be unique. Provide the date column as "yyyy-mm-dd".
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.

Value

A sentocorpus object, as a quanteda classed list keeping the elements documents, metadata andsettings, while adding the elements features (a character vector of all features part of the corpus.), and nDocs (the number of documents in the corpus).

See Also

corpus

Examples

```
# corpus construction
data("useconomynews")
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	a sentiment scores data.table with a date and feature–lexicon 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.

See Also

compute_sentiment, perform_agg

Examples

```
# computation of sentiment measures based on a provided control function
data("useconomynews")
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)
```

---

sento_model *Optimized and automated sparse regression*

Description

Linear or nonlinear penalized regression of a 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 forecasting performance metrics. Independent variables are normalized in the regression process, but coefficients are returned in their original space.

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 last factor level or column as the reference class. No `NA` values are allowed.
`x`	a named `data.frame` with other explanatory variables, by default set to `NULL`.
`ctr`	output from a `ctr_model()` call.

Value

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

`reg`	optimized regression, i.e. a model-specific `glmnet` object.
`sentomeasures`	the input `sentomeasures` object.
`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"`).

If `ctr$do.iter == TRUE`, a list containing:

`regs`	optimized 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.
`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), `"MAPE"` (mean absolute percentage error), `"DA"` (directional accuracy), `"accuracy"` (proportion of correctly predicted classes in case of a logistic regression), and each's respective individual values in the sample. Only the relevant performance statistics are given depending on the type of regression. Dates are similarly as with the `"regs"` output element from the perspective of the sentiment measures.

See Also

`ctr_model`, `glmnet`, `train`

Examples

```
# construct a sentomeasures object to start with
data("useconomynews")
useconomynews <- useconomynews[date >= "1988-01-01", ]
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
data("sp500")
y <- sp500$return # convert to numeric vector
sentomeasures <- fill_measures(sentomeasures)
length(y) == nrow(sentomeasures$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 = "lm", type = ic, do.iter = FALSE, h = 0)
   out1[[ic]] <- sento_model(sentomeasures, y, x = x, ctr = ctrIC)
}

# a (very) short iterative analysis of cross-validation based models
ctrCV <- ctr_model(model = "lm", type = "cv", do.iter = TRUE, h = 0, trainWindow = 250,
                   testWindow = 20, oos = 0, nSample = 320, do.progress = TRUE)
out2 <- sento_model(sentomeasures, y, x = x, ctr = ctrCV)

# a similar iterative analysis of cross-validation based models but for a binomial target
yb <- sp500$up
ctrCV <- ctr_model(model = "binomial", type = "cv", do.iter = TRUE, trainWindow = 250,
                   h = 0, testWindow = 20, oos = 0, nSample = 320, do.progress = TRUE)
out3 <- sento_model(sentomeasures, yb, x = x, ctr = ctrCV)

# post-analysis (summary, attribution and prediction)
out <- out1[["BIC"]]
summary(out)

attribution <- retrieve_attributions(out)

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

`setup_lexicons` *Setup 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 **quanteda** 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 `data.table` list element according to its name, 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.

See Also

`as_key`

Examples

```
# 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.frame(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.

- date. Date as `"yyyy-mm-01"`.
- return. A `numeric` monthly return value.
- up. A `factor` with value `"pos"` if 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

`summary.sentomodel` *Summary of a sentomodel object*

Description

Prints out a short summary consisting of the main elements of the constructed model (model type, calibrated values, non-zero coefficients and performance).

Usage

```
## S3 method for class 'sentomodel'
summary(object, ...)
```

Arguments

`object`	a `sentomodel` object.
`...`	not used.

Value

A sequence of informative prints on the model's results.

See Also

`sento_model`

`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.
- headline. Headlines 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