Package 'sentometrics'

September 12, 2019

Type Package

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

Version 0.7.0

Author Samuel Borms [aut, cre],
David Ardia [aut],
Keven Bluteau [aut],
Kris Boudt [aut],
Jeroen Van Pelt [ctb],
Andres Algaba [ctb]

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

Description Optimized prediction based on textual sentiment, accounting for the intrinsic challenge that sentiment can be computed and pooled across texts and time in various ways. See Ardia et al. (2018) <doi:10.2139/ssrn.3067734>.

Depends R (>= 3.3.0), data.table

License GPL (>= 2)

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

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

Encoding UTF-8

LazyData true

Suggests covr, doParallel, e1071, parallel, randomForest, testthat, tm, NLP, shiny

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

LinkingTo Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 6.1.1

SystemRequirements GNU make

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-09-12 09:30:02 UTC

R topics documented:

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

Description

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

Main functions

- Corpus (features) generation: `sento_corpus`, `add_features`, `as.sento_corpus`
- Sentiment computation and aggregation into sentiment measures: `ctr_agg`, `sento_lexicons`, `compute_sentiment`, `aggregate.sentiment`, `peakdocs`, `sento_measures`, `peakdates`, `aggregate.sento_measure`
  and a series of manipulation functions
- Sparse modelling: `ctr_model`, `sento_model`
- Prediction and post-modelling analysis: `predict.sento_model`, `attributions`

The `sento_app` function is a Shiny interface for fast document-level sentiment computation and aggregation, and downloading of the obtained values. It serves as a visualisation of part of what the package does.

Note

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

Author(s)

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

Authors:

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

Other contributors:

- Jeroen Van Pelt <jeroenvanpelt@hotmail.com> [contributor]
- Andres Algaba <andres.algaba@vub.be> [contributor]

References

Ardia, Bluteau and Boudt (2019). "Questioning the news about economic growth: Sparse forecasting using thousands of news-based sentiment values". *International Journal of Forecasting, forthcoming*, https://doi.org/10.2139/ssrn.2976084.

Ardia, Bluteau, Borms and Boudt (2018). "The R package sentometrics to compute, aggregate and predict with textual sentiment". *Working paper*, https://doi.org/10.2139/ssrn.3067734.

See Also

Useful links:

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

---

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

Description

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

Usage

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

Arguments

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

do.regex — a logical vector equal in length to the number of elements in the keywords argument list, or a single value if it applies to all. It should be set to TRUE at those positions where a single regex expression is used to identify the particular feature.

Details

If a provided feature name is already part of the corpus, it will be replaced. The featuresdf and keywords arguments can be provided at the same time, or only one of them, leaving the other at NULL. We use the **stringi** package for searching the keywords. The do.regex argument points to the corresponding elements in keywords. For FALSE, we transform the keywords into a simple regex expression, involving "\b" for exact word boundary matching and (if multiple keywords) | as OR operator. The elements associated to TRUE do not undergo this transformation, and are evaluated as given, if the corresponding keywords vector consists of only one expression. For a large corpus and/or complex regex patterns, this function may require some patience. Scaling between 0 and 1 is performed via min-max normalization, per column.

Value

An updated corpus object.

Author(s)

Samuel Borms

Examples

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

set.seed(505)

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

sum(quanteda::docvars(corpus3, "pres")) ==
  sum(quanteda::docvars(corpus4, "pres2")) # TRUE

# adding a complementary feature
```

```
nonpres <- data.frame(nonpres = as.numeric(!quanteda::docvars(corpus3, "pres")))
corpus3 <- add_features(corpus3, featuresdf = nonpres)
```

`aggregate.sentiment` *Aggregate textual sentiment across sentences, documents and time*

Description

Aggregates textual sentiment scores at sentence- or document-level into a panel of textual sentiment measures. Can also be used to aggregate sentence-level sentiment scores into document-level sentiment scores. This function is called within the `sento_measures` function.

Usage

```
## S3 method for class 'sentiment'
aggregate(x, ctr, do.full = TRUE, ...)
```

Arguments

`x`	a `sentiment` object created using `compute_sentiment` (from a `sento_corpus` object), or an output from `to_sentiment`.
`ctr`	output from a `ctr_agg` call. The `howWithin` and `nCore` elements are ignored.
`do.full`	if `do.full` = `TRUE` (by default), does entire aggregation up to a `sento_measures` object, else only goes from sentence-level to document-level. Ignored if no `"sentence_id"` column in `sentiment` input object.
`...`	not used.

Value

A document-level `sentiment` object or a fully aggregated `sento_measures` object.

Author(s)

Samuel Borms, Keven Bluteau

See Also

`compute_sentiment`, `ctr_agg`, `sento_measures`

Examples

```
set.seed(505)

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

# computation of sentiment
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l1 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")],
                     list_valence_shifters[["en"]])
l2 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")],
                     list_valence_shifters[["en"]][, c("x", "t")])
sent1 <- compute_sentiment(corpusSample, l1, how = "counts")
sent2 <- compute_sentiment(corpusSample, l2, how = "counts", do.sentence = TRUE)
ctr <- ctr_agg(howTime = c("linear"), by = "year", lag = 3)

# aggregate into sentiment measures
sm1 <- aggregate(sent1, ctr)
sm2 <- aggregate(sent2, ctr)

# two-step aggregation (first into document-level sentiment)
sent3 <- aggregate(sent2, ctr, do.full = FALSE)
sm3 <- aggregate(sent3, ctr)
```

---

aggregate.sento_measures

Aggregate sentiment measures

Description

Aggregates sentiment measures by combining across provided lexicons, features, and time weighting schemes dimensions. The combination occurs by taking the mean of the relevant measures.

Usage

```
## S3 method for class 'sento_measures'
aggregate(x, features = NULL, lexicons = NULL,
  time = NULL, do.keep = FALSE, ...)
```

Arguments

x	a `sento_measures` object created using `sento_measures`.
features	a `list` with unique features to aggregate at given name, e.g., `list(feat12 = c("feat1","feat2"))`. See `sento_measures$features` for the exact names to use. Use `NULL` (default) to apply no merging across this dimension.

lexicons	a list with unique lexicons to aggregate at given name, e.g., list(lex12 = c("lex1","lex2")). See sento_measures$lexicons for the exact names to use. Use NULL (default) to apply no merging across this dimension.
time	a list with unique time weighting schemes to aggregate at given name, e.g., list(tw12 = c("tw1","tw2")). See sento_measures$time for the exact names to use. Use NULL (default) to apply no merging across this dimension.
do.keep	a logical indicating if the original sentiment measures should be kept (i.e., the aggregated sentiment measures will be added to the current sentiment measures as additional indices if do.keep = TRUE).
...	not used.

Value

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

Author(s)

Samuel Borms

Examples

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

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

# aggregation across specified components
smAgg <- aggregate(sento_measures,
                   time = list(W = c("equal_weight", "linear")),
                   features = list(journals = c("wsj", "wapo")),
                   do.keep = TRUE)

# aggregation in full
dims <- get_dimensions(sento_measures)
smFull <- aggregate(sento_measures,
                    lexicons = list(L = dims[["lexicons"]]),
                    time = list(T = dims[["time"]]),
                    features = list(F = dims[["features"]]))

## Not run:
# aggregation won't work, but produces informative error message
aggregate(sento_measures,
          time = list(W = c("equal_weight", "almon1")),
```

```
          lexicons = list(LEX = c("LM_en")),
          features = list(journals = c("notInHere", "wapo")))
## End(Not run)
```

`as.data.table.sento_measures` *Get the sentiment measures*

Description

Extracts the sentiment measures `data.table` in either wide (by default) or long format.

Usage

```
## S3 method for class 'sento_measures'
as.data.table(x, keep.rownames = FALSE,
  format = "wide", ...)
```

Arguments

`x`	a `sento_measures` object created using `sento_measures`.
`keep.rownames`	see `as.data.table`.
`format`	a single `character` vector, one of `c("wide","long")`.
`...`	not used.

Value

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

Author(s)

Samuel Borms

Examples

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

sm <- sento_measures(sento_corpus(corpusdf = usnews[1:200, ]),
                     sento_lexicons(list_lexicons["LM_en"]),
                     ctr_agg(lag = 3))

as.data.table(sm)
as.data.table(sm, "long")
```

as.sentiment *Convert a sentiment table to a sentiment object*

Description

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

Usage

```
as.sentiment(s)
```

Arguments

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

Value

A sentiment object.

Author(s)

Samuel Borms

Examples

```
set.seed(505)

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

ids <- paste0("id", 1:200)
dates <- sample(seq(as.Date("2015-01-01"), as.Date("2018-01-01"), by = "day"), 200, TRUE)
word_count <- sample(150:850, 200, replace = TRUE)
sent <- matrix(rnorm(200 * 8), nrow =  200)
s1 <- s2 <- data.table(id = ids, date = dates, word_count = word_count, sent)
s3 <- data.frame(id = ids, date = dates, word_count = word_count, sent,
                 stringsAsFactors = FALSE)
s4 <- compute_sentiment(usnews$texts[201:400],
                        sento_lexicons(list_lexicons["GI_en"]),
                        "counts", do.sentence = TRUE)
m <- "method"
```

```
colnames(s1)[-c(1:3)] <- paste0(m, 1:8)
sent1 <- as.sentiment(s1)

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

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

s4[, "date" := rep(dates, s4[, max(sentence_id), by = id][[2]])]
sent4 <- as.sentiment(s4)

# further aggregation from then on is easy...
sentMeas1 <- aggregate(sent1, ctr_agg(lag = 10))
sent5 <- aggregate(sent4, ctr_agg(howDocs = "proportional"), do.full = FALSE)
```

`as.sento_corpus` *Convert a quanteda or tm corpus object into a sento_corpus object*

Description

Converts most common **quanteda** and **tm** corpus objects into a `sento_corpus` object. Appropriate available metadata is integrated as features; for a **quanteda** corpus, this can come from `docvars(x)`, for a **tm** corpus, only `meta(x, type = "indexed")` metadata is considered.

Usage

```
as.sento_corpus(x, dates = NULL, do.clean = FALSE)
```

Arguments

`x`	a **quanteda** `corpus` object, a **tm** `SimpleCorpus` or a **tm** `VCorpus` object. For **tm** corpora, every corpus element should consist of a single `"content"` `character` vector as the document unit.
`dates`	an optional sequence of dates as `"yyyy-mm-dd"`, of the same length as the number of documents in the input corpus, to define the `"date"` column. If `dates = NULL`, the `"date"` metadata element in the input corpus, if available, will be used but should be in the same `"yyyy-mm-dd"` format.
`do.clean`	see `sento_corpus`.

Value

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

Author(s)

Samuel Borms

See Also

corpus, SimpleCorpus, VCorpus, sento_corpus

Examples

```
data("usnews", package = "sentometrics")
txt <- system.file("texts", "txt", package = "tm")
reuters <- system.file("texts", "crude", package = "tm")

# reshuffle usnews data.frame for use in quanteda and tm
dates <- usnews$date
usnews$wrong <- "notNumeric"
colnames(usnews)[c(1, 3)] <- c("doc_id", "text")

# conversion from a quanteda corpus
qcorp <- quanteda::corpus(usnews,
                          text_field = "text", docid_field = "doc_id")
corp1 <- as.sento_corpus(qcorp)
corp2 <- as.sento_corpus(qcorp, sample(dates)) # overwrites "date" column

# conversion from a tm SimpleCorpus corpus (DataframeSource)
tmSCdf <- tm::SimpleCorpus(tm::DataframeSource(usnews))
corp3 <- as.sento_corpus(tmSCdf)

# conversion from a tm SimpleCorpus corpus (DirSource)
tmSCdir <- tm::SimpleCorpus(tm::DirSource(txt))
corp4 <- as.sento_corpus(tmSCdir, dates[1:length(tmSCdir)])

# conversion from a tm VCorpus corpus (DataframeSource)
tmVCdf <- tm::VCorpus(tm::DataframeSource(usnews))
corp5 <- as.sento_corpus(tmVCdf)

# conversion from a tm VCorpus corpus (DirSource)
tmVCdir <- tm::VCorpus(tm::DirSource(reuters),
                       list(reader = tm::readReut21578XMLasPlain))
corp6 <- as.sento_corpus(tmVCdir, dates[1:length(tmVCdir)])
```

---

attributions *Retrieve top-down model sentiment attributions*

Description

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

Usage

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

Arguments

model	a sento_model or a sento_modelIter object created with sento_model.
sento_measures	the sento_measures object, as created with sento_measures, used to estimate the model from the first argument (make sure this is the case!).
do.lags	a logical, TRUE also computes the attribution to each time lag. For large time lags, this is time-consuming.
do.normalize	a logical, TRUE divides each element of every attribution vector at a given date by its L2-norm at that date, normalizing the values between -1 and 1. The document attributions are not normalized.
refDates	the dates (as "yyyy-mm-dd") at which attribution is to be performed. These should be between the latest date available in the input sento_measures object and the first estimation sample date (that is, model$dates[1] if model is a sento_model object). All dates should also be in get_dates(sento_measures). If NULL (default), attribution is calculated for all in-sample dates. Ignored if model is a sento_modelIter object, for which attribution is calculated for all out-of-sample prediction dates.
factor	the factor level as a single character vector for which attribution has to be calculated in case of (a) multinomial model(s). Ignored for linear and binomial models.

Details

See sento_model for an elaborate modelling example including the calculation and plotting of attributions. The attribution for logistic models is represented in terms of log odds. For binomial models, it is calculated with respect to the last factor level or factor column. A NULL value for document-level attribution on a given date means no documents are directly implicated in the associated prediction.

Value

A list of class attributions, with "documents", "lags", "lexicons", "features" and "time" as dimensions for which aggregation is computed. The last four dimensions are data.tables having a "date" column and the other columns the different components of the dimension, with the attributions as values. Document-level attribution is further decomposed into a data.table per date, with "id", "date" and "attrib" columns. If do.lags = FALSE, the "lags" element is set to NULL.

Author(s)

Samuel Borms, Keven Bluteau

See Also

sento_model

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

Description

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

Usage

```
compute_sentiment(x, lexicons, how = "proportional", tokens = NULL,
  do.sentence = FALSE, nCore = 1)
```

Arguments

x	either a sento_corpus object created with sento_corpus, a **quanteda** corpus object, a **tm** SimpleCorpus object, a **tm** VCorpus object, or a character vector. Only a sento_corpus object incorporates a date dimension. In case of a corpus object, the numeric columns from the docvars are considered as features over which sentiment will be computed. In case of a character vector, sentiment is only computed across lexicons.
lexicons	a sento_lexicons object created using sento_lexicons.
how	a single character vector defining how aggregation within documents should be performed. For currently available options on how aggregation can occur, see get_hows()$words.
tokens	a list of tokenized documents, or if do.sentence = TRUE a list of a list of tokenized sentences. This allows to specify your own tokenization scheme. Can result from the **quanteda**'s tokens function, the **tokenizers** package, or other. Make sure the tokens are constructed from (the texts from) the x argument, are unigrams, and preferably set to lowercase, otherwise, results may be spurious and errors could occur. By default set to NULL.
do.sentence	a logical to indicate whether the sentiment computation should be done on sentence-level rather than document-level. By default do.sentence = FALSE. The methodology defined in the **sentimentr** package is followed to carry out the computation.
nCore	a positive numeric that will be passed on to the numThreads argument of the setThreadOptions function, to parallelize the sentiment computation across texts. A value of 1 (default) implies no parallelization. Parallelization is expected to improve speed of the sentiment computation only for sufficiently large corpora.

Details

For a separate calculation of positive (resp. negative) sentiment, one has to provide distinct positive (resp. negative) lexicons. This can be done using the `do.split` option in the `sento_lexicons` function, which splits out the lexicons into a positive and a negative polarity counterpart. All `NA`s are converted to 0, under the assumption that this is equivalent to no sentiment. If `tokens = NULL` (as per default), texts are tokenized as unigrams. Punctuation and numbers are removed, but not stopwords. The number of words for each document is computed based on that same tokenization. All tokens are converted to lowercase, in line with what the `sento_lexicons` function does for the lexicons and valence shifters.

Value

If `x` is a `sento_corpus` object, a `sentiment` object, i.e., a `data.table` containing the sentiment scores `data.table` with an `"id"`, a `"date"` and a `"word_count"` column, and all lexicon-feature sentiment scores columns. A `sentiment` object can be used for aggregation into time series with the `aggregate.sentiment` function. If `do.sentence = TRUE`, an additional `"sentence_id"` column along the `"id"` column is added.

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

If `x` is a **tm** `SimpleCorpus` object, a **tm** `VCorpus` object, or a `character` vector, a sentiment scores `data.table` with an auto-created `"id"` column, a `"word_count"` column, and all lexicon sentiment scores columns.

Calculation

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

The sentiment calculation on sentence level approaches each sentence as if it is a document. Depending on the input either the unigrams, bigrams or clusters approach is used. For the latter, we replicated exactly the default **sentimentr** package method, with a cluster of five words before and two words after the polarized word. If there are commas around the polarized word, the cluster is limited (extended) to the words after the previous comma and before the next comma. Adversative conjunctions (adv, `t = 4`) are accounted for here. If the value $1 + 0.25adv$ is greater (resp. smaller) than 1, it is added to (resp. substracted from) the total amplification weight.

The how = "proportionalPol" option divides each document's sentiment score by the number of detected polarized words (counting words that appear multiple times by their frequency), instead of the total number of words which the how = "proportional" option gives. The how = "counts" option does no normalization. The squareRootCounts divides the sentiment by the square root of the number of tokens in each text. The how = "UShaped" option gives a higher weight to words at the beginning and end of the texts. The how = "invertedUShaped" option gives a lower weight to words at the beginning and the end of the texts. The how = "exponential" option gives gradually more weight the later the word appears in the text. The how = "invertedExponential" option gives gradually less weight the later the words appears in the text. The how = "TF" option gives a weight proportional to the number of times a word appears in a text. The how = "logarithmicTF" option gives the same weight as "TF" but logarithmically scaled. The how = "augmentedTF" option can be used to prevent a bias towards longer documents. The weight is determined by dividing the raw frequency of a token by the raw frequency of the most occurring term in the document. The how = "IDF" option uses the logarithm of the division of the raw frequency of a word by the number of texts in which the word appears. By doing this, words appearing in multiple texts get a lower weight. The how = "TFIDF", how = "logarithmicTFIDF" and how = "augmentedTFIDF" options use the same weights as their IDF-variant but then multiplied with the how = "IDF" option. See the vignette for more details.

Author(s)

Samuel Borms, Jeroen Van Pelt, Andres Algaba

Examples

```
data("usnews", package = "sentometrics")
txt <- system.file("texts", "txt", package = "tm")
reuters <- system.file("texts", "crude", package = "tm")
data("list_lexicons", package = "sentometrics")
data("list_valence_shifters", package = "sentometrics")

l1 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
l2 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")],
                     list_valence_shifters[["en"]])
l3 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")],
                     list_valence_shifters[["en"]][, c("x", "t")])

# from a sento_corpus object - unigrams approach
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 200)
sent1 <- compute_sentiment(corpusSample, l1, how = "proportionalPol")

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

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

# from an already tokenized corpus - using the 'tokens' argument
```

```
toks <- as.list(quanteda::tokens(corpusQSample, what = "fastestword"))
sent4 <- compute_sentiment(corpusQSample, l1[1], how = "counts", tokens = toks)

# from a SimpleCorpus object - unigrams approach
scorp <- tm::SimpleCorpus(tm::DirSource(txt, encoding = "UTF-8"))
sent5 <- compute_sentiment(scorp, l1, how = "proportional")

# from a VCorpus object - unigrams approach
## in contrast to what as.sento_corpus(vcorp) would do, the
## sentiment calculator handles multiple character vectors within
## a single corpus element as separate documents
vcorp <- tm::VCorpus(tm::DirSource(reuters))
sent6 <- compute_sentiment(vcorp, l1, how = "proportional")

# from a sento_corpus object - unigrams approach with tf-idf weighting
sent7 <- compute_sentiment(corpusSample, l1, how = "TFIDF")

# sentence-by-sentence computation
sent8 <- compute_sentiment(corpusSample, l1, how = "squareRootCounts",
                           do.sentence = TRUE)

# from an artificially constructed multilingual corpus
usnews[["language"]] <- "en" # add language column
usnews$language[1:100] <- "fr"
l_en <- sento_lexicons(list("FEEL_en" = list_lexicons$FEEL_en_tr))
l_fr <- sento_lexicons(list("FEEL_fr" = list_lexicons$FEEL_fr))
lexicons <- list(en = l_en, fr = l_fr)
corpusLang <- sento_corpus(corpusdf = usnews[1:250, ])
sent9 <- compute_sentiment(corpusLang, lexicons, how = "proportional")
```

`corpus_summarize` *Summarize the sento_corpus object*

Description

Summarizes the `sento_corpus` object and returns insights about features and tokens over time.

Usage

```
corpus_summarize(x, by = "day", features = NULL)
```

Arguments

`x`	is a `sento_corpus` object created with `sento_corpus`
`by`	a single `character` vector to specify the frequency time interval over which the statistics need to be calculated.
`features`	a `character` vector that can be used to select a subset of the features to be analysed.

Details

This function summarizes the sento_corpus object by generating statistics about features and tokens over time. The insights can be narrowed down to a chosen set of metadata features. The same tokenization as in the sentiment calculation in compute_sentiment is used.

Value

returns a list containing:

stats	a data.table with statistics about the number of documents, total, average, minimum and maximum number of tokens and the number of texts per features for each date.
plots	a list with three plots representing the above statistics.

Author(s)

Jeroen Van Pelt, Samuel Borms, Andres Algaba

Examples

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

corpus <- sento_corpus(usnews)

# summary of corpus by day
summary1 <- corpus_summarize(corpus)

# summary of corpus by month
summary2 <- corpus_summarize(corpus, by = "month")
```

---

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.sentence = FALSE,
  do.ignoreZeros = TRUE, by = "day", lag = 1, fill = "zero",
  alphaExpDocs = 0.1, alphasExp = seq(0.1, 0.5, by = 0.1),
  ordersAlm = 1:3, do.inverseAlm = TRUE, aBeta = 1:4, bBeta = 1:4,
  weights = NULL, tokens = NULL, nCore = 1)
```

Arguments

howWithin	a single character vector defining how aggregation within documents will be performed. Should length(howWithin) > 1, the first element is used. For available options on how this aggregation can occur; see get_hows()$words.
howDocs	a single character vector defining how aggregation across documents (and/or sentences) per date will be performed. Should length(howDocs) > 1, the first element is used. For available options on how this aggregation can occur; see get_hows()$docs.
howTime	a character vector defining how aggregation across dates will be performed. More than one choice is possible. For available options on how this aggregation can occur; see get_hows()$time.
do.sentence	see compute_sentiment.
do.ignoreZeros	a logical indicating whether zero sentiment values have to be ignored in the determination of the document (and/or sentence) weights while aggregating across documents (and/or sentences). By default do.ignoreZeros = TRUE, such that documents (and/or sentences) with a raw sentiment score of zero or for which a given feature indicator is equal to zero are considered irrelevant.
by	a single character vector, either "day","week","month" or "year", to indicate at what level the dates should be aggregated. Dates are displayed as the first day of the period, if applicable (e.g., "2017-03-01" for March 2017).
lag	a single integer vector, being the time lag to be specified for aggregation across time. By default equal to 1, meaning no aggregation across time; a time weighting scheme named "dummyTime" is used in this case.
fill	a single character vector, one of c("zero","latest","none"), to control how missing sentiment values across the continuum of dates considered are added. This impacts the aggregation across time, applying the measures_fill function before aggregating, except if fill = "none". By default equal to "zero", which sets the scores (and thus also the weights) of the added dates to zero in the time aggregation.
alphaExpDocs	a single integer vector. A weighting smoothing factor, used if "exponential" %in% howDocs or "inverseExponential" %in% howDocs. Value should be between 0 and 1 (both excluded); see weights_exponential.
alphasExp	a numeric vector of all exponential weighting smoothing factors, used if "exponential" %in% howTime. Values should be between 0 and 1 (both excluded); see weights_exponential.
ordersAlm	a numeric vector of all Almon polynomial orders (positive) to calculate weights for, used if "almon" %in% howTime; see weights_almon.
do.inverseAlm	a logical indicating if for every Almon polynomial its inverse has to be added, used if "almon" %in% howTime; see weights_almon.
aBeta	a numeric vector of positive values as first Beta weighting decay parameter; see weights_beta.
bBeta	a numeric vector of positive values as second Beta weighting decay parameter; see weights_beta.

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

Details

For currently available options on how aggregation can occur (via the howWithin, howDocs and howTime arguments), call get_hows. The control parameters associated to howDocs are used both for aggregation across documents and across sentences.

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

measures_fill, almons, compute_sentiment

Examples

```
set.seed(505)

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

# more elaborate control function (particular attention to time weighting schemes)
ctr2 <- ctr_agg(howWithin = "proportionalPol",
                howDocs = "exponential",
             howTime = c("equal_weight", "linear", "almon", "beta", "exponential", "own"),
                do.ignoreZeros = TRUE,
                by = "day",
                lag = 20,
                ordersAlm = 1:3,
                do.inverseAlm = TRUE,
                alphasExp = c(0.20, 0.50, 0.70, 0.95),
                aBeta = c(1, 3),
                bBeta = c(1, 3, 4, 7),
                weights = data.frame(myWeights = runif(20)),
                alphaExp = 0.3)

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

ctr_model *Set up control for sentiment-based sparse regression modelling*

Description

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

Usage

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

Arguments

model	a character vector with one of the following: "gaussian" (linear regression), "binomial" (binomial logistic regression), or "multinomial" (multinomial logistic regression).
type	a character vector indicating which model calibration approach to use. Supports "BIC", "AIC" and "Cp" (Mallows's Cp) as sparse regression adapted information criteria (Tibshirani and Taylor, 2012; Zou, Hastie and Tibshirani, 2007), and "cv" (cross-validation based on the train function from the **caret** package). The adapted information criteria are only available for a linear regression.
do.intercept	a logical, TRUE by default fits an intercept.
do.iter	a logical, TRUE induces an iterative estimation of models at the given nSample size and performs the associated out-of-sample prediction exercise through time.
h	an integer value that shifts the time series to have the desired prediction setup; h = 0 means no change to the input data (nowcasting assuming data is aligned properly), h > 0 shifts the dependent variable by h periods (i.e., rows) further in time (forecasting), h < 0 shifts the independent variables by h periods.
oos	a non-negative integer to indicate the number of periods to skip from the end of the training sample up to the out-of-sample prediction(s). This is either used in the cross-validation based calibration approach (if type = "cv"), or for the iterative out-of-sample prediction analysis (if do.iter = TRUE). For instance, given t, the (first) out-of-sample prediction is computed at $t+$ oos $+1$.
do.difference	a logical, TRUE will difference the target variable y supplied in the sento_model function with as lag the absolute value of the h argument, but abs(h) > 0 is required. For example, if h = 2, and assuming the y variable is properly aligned date-wise with the explanatory variables denoted by X (the sentiment measures

and other in x), the regression will be of $y_{t+2} - y_t$ on X_t. If h = -2, the regression fitted is $y_{t+2} - y_t$ on X_{t+2}. The argument is always kept at FALSE if the model argument is one of c("binomial","multinomial").

alphas — a numeric vector of the alphas to test for during calibration, between 0 and 1. A value of 0 pertains to Ridge regression, a value of 1 to LASSO regression; values in between are pure elastic net.

lambdas — a numeric vector of the lambdas to test for during calibration, greater or equal than zero. A value of zero means no regularization, thus requires care when the data is fat. By default set to NULL, such that the lambdas sequence is generated by the glmnet function or set to 10^seq(2,-2,length.out = 100) in case of cross-validation.

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

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

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

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

do.shrinkage.x — a logical vector to indicate which of the other regressors provided through the x argument of the sento_model function should be subject to shrinkage (TRUE). If argument is of length one, it applies to all external regressors.

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

nCore — a positive integer to indicate the number of cores to use for a parallel iterative model estimation (do.iter = TRUE). We use the %dopar% construct from the **foreach** package. By default, nCore = 1, which implies no parallelization. No progress statements are displayed whatsoever when nCore > 1. For cross-validation models, parallelization can also be carried out for a single-shot model (do.iter = FALSE), whenever a parallel backend is set up. See the examples in sento_model.

Value

A list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

References

Tibshirani and Taylor (2012). "Degrees of freedom in LASSO problems". *The Annals of Statistics 40, 1198-12*, https://doi.org/10.1214/12-AOS1003.

Zou, Hastie and Tibshirani (2007). "On the 'degrees of freedom' of the LASSO". *The Annals of Statistics 35, 2173-2192*, https://doi.org/10.1214/009053607000000127.

See Also

sento_model

Examples

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

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

---

diff.sento_measures *Differencing of sentiment measures*

Description

Differences the sentiment measures from a sento_measures object.

Usage

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

Arguments

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

Value

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

Author(s)

Samuel Borms

Examples

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

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

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

---

epu *Monthly Economic Policy Uncertainty Index*

Description

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

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

Usage

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

Format

A `data.frame` with 403 rows and 4 columns.

Source

Measuring Economic Policy Uncertainty. Retrieved August 24, 2018.

References

Baker, Bloom and Davis (2016). "Measuring Economic Policy Uncertainty". *The Quarterly Journal of Economics 131, 1593-1636*, https://doi.org/10.1093/qje/qjw024.

Examples

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

`get_dates` *Get the dates of the sentiment measures/time series*

Description

Returns the dates of the sentiment time series.

Usage

```
get_dates(sento_measures)
```

Arguments

`sento_measures`	a `sento_measures` object created using `sento_measures`.

Value

The `"date"` column in `sento_measures[["measures"]]` as a `character` vector.

Author(s)

Samuel Borms

`get_dimensions` *Get the dimensions of the sentiment measures*

Description

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

Usage

```
get_dimensions(sento_measures)
```

Arguments

`sento_measures`	a `sento_measures` object created using `sento_measures`.

Value

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

Author(s)

Samuel Borms

---

get_hows *Options supported to perform aggregation into sentiment measures*

Description

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

Usage

```
get_hows()
```

Details

See the package's vignette for a detailed explanation of all aggregation options.

Value

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

See Also

ctr_agg

---

get_loss_data *Retrieve loss data from a selection of models*

Description

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

Usage

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

Arguments

models	a named list of sento_modelIter 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), "error" (prediction minus realized response variable), "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 four options are available for "gaussian" models, the last option applies only to "binomial" and "multinomial" models.

Value

A matrix of loss data.

Author(s)

Samuel Borms

References

Hansen, Lunde and Nason (2011). "The model confidence set". *Econometrica 79, 453-497*, https://doi.org/10.3982/ECTA5771.

See Also

sento_model, MCSprocedure

Examples

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

set.seed(505)

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

ctrA <- ctr_agg(howWithin = "proportionalPol", howDocs = "proportional",
                howTime = c("equal_weight", "linear"), by = "month", lag = 3)
sentMeas <- sento_measures(corpus, l, ctrA)

# prepare y and other x variables
y <- epu[epu$date %in% get_dates(sentMeas), "index"]
length(y) == nobs(sentMeas) # TRUE
```

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

# estimate different type of regressions
ctrM <- ctr_model(model = "gaussian", type = "AIC", do.iter = TRUE,
                  h = 0, nSample = 120, start = 50)
out1 <- sento_model(sentMeas, y, x = x, ctr = ctrM)
out2 <- sento_model(sentMeas, y, x = NULL, ctr = ctrM)
out3 <- sento_model(measures_select(sentMeas, "linear"), y, x = x, ctr = ctrM)
out4 <- sento_model(measures_select(sentMeas, "linear"), y, x = NULL, ctr = ctrM)

lossData <- get_loss_data(models = list(m1 = out1, m2 = out2, m3 = out3, m4 = out4),
                          loss = "errorSq")

mcs <- MCS::MCSprocedure(lossData)
## End(Not run)
```

`list_lexicons` *Built-in lexicons*

Description

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

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

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

Usage

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

Format

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

Source

FEEL lexicon. Retrieved November 1, 2017.

GI lexicon. Retrieved November 1, 2017.

HENRY lexicon. Retrieved November 1, 2017.

LM lexicon. Retrieved November 1, 2017.

References

Abdaoui, Azé, Bringay and Poncelet (2017). "FEEL: French Expanded Emotion Lexicon". *Language Resources & Evaluation 51, 833-855*, https://doi.org/10.1007/s10579-016-9364-5.

Henry (2008). "Are investors influenced by how earnings press releases are written?". *Journal of Business Communication 45, 363-407*, https://doi.org/10.1177/0021943608319388.

Loughran and McDonald (2011). "When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks". *Journal of Finance 66, 35-65*, https://doi.org/10.1111/j.1540-6261.2010.01625.x.

Examples

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

`list_valence_shifters` *Built-in valence word lists*

Description

A `list` containing all built-in valence word lists, as `data.tables` with three columns: a `x` column with the words, a `y` column with the values associated to each word, and a `t` column with the type of valence shifter (`1` = negators, `2` = amplifiers, `3` = deamplifiers). The `list` element names indicate the language (based on the two-letter ISO code convention as in `stopwords`) of the valence word list. All non-English word lists are translated via Microsoft Translator through Microsoft Word. Only the entries whose translation differs from the original entry are kept. All words are unigrams and in lowercase. The built-in valence word lists are available in following languages:

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

Usage

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

Format

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

Source

hash_valence_shifters (English valence shifters). Retrieved August 24, 2018.

Examples

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

---

measures_fill *Add and fill missing dates to sentiment measures*

Description

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

Usage

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

Arguments

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

Details

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

Value

A modified sento_measures object.

Author(s)

Samuel Borms

Examples

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

# construct a sento_measures object to start with
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "day", lag = 7, fill = "none")
sento_measures <- sento_measures(corpusSample, l, ctr)

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

---

measures_global *Aggregate sentiment measures into multiple weighted global sentiment indices*

Description

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

Usage

```
measures_global(sento_measures, lexicons = 1, features = 1, time = 1)
```

Arguments

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

Details

This particular function returns no new sento_measures object. The measures are constructed from weights that indicate the importance (and sign) along each component from the lexicons, features, and time dimensions. There is no restriction in terms of allowed weights. For example, the global index based on the supplied lexicon weights ("globLex") is obtained first by multiplying every sentiment measure with its corresponding weight (meaning, the weight given to the lexicon the sentiment is computed with), then by taking the average per date.

Value

A data.table with the different types of weighted global sentiment measures, named "globLex", "globFeat", "globTime" and "global", with "date" as the first column. The last measure is an average of the the three other measures.

Author(s)

Samuel Borms

See Also

sento_model

Examples

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

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

# aggregate into one global sentiment measure given a weighting for lexicons and features
global <- measures_global(sento_measures,
                          lexicons = c(0.40, 0.60),
                          features = c(0.10, -0.20, 0.30, -1),
                          time = 1)
```

---

measures_update *Update sentiment measures*

Description

Updates a sento_measures object based on a new corpus provided. Sentiment for the unseen corpus texts calculated and aggregated applying the control variables from the input sento_measures object.

Usage

```
measures_update(sento_measures, sento_corpus, lexicons)
```

Arguments

sento_measures	sento_measures object created with sento_measures
sento_corpus	a sento_corpus object created with sento_corpus.
lexicons	a sento_lexicons object created with sento_lexicons.

Value

An updated sento_measures object.

Author(s)

Jeroen Van Pelt, Samuel Borms, Andres Algaba

See Also

sento_measures, compute_sentiment

Examples

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

corpus1 <- sento_corpus(usnews[1:500, ])
corpus2 <- sento_corpus(usnews[400:2000, ])

ctr <- ctr_agg(howTime = "linear", by = "year", lag = 3)
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")],
                    list_valence_shifters[["en"]])
sento_measures <- sento_measures(corpus1, l, ctr)
sento_measuresNew <- measures_update(sento_measures, corpus2, l)
```

---

merge.sentiment *Merge sentiment objects horizontally or vertically row-wise*

Description

Combines multiple sentiment objects with possibly different column names into a new sentiment object. Here, too, any resulting NA values are converted to zero.

Usage

```
## S3 method for class 'sentiment'
merge(...)
```

Arguments

...	sentiment objects to merge.

Value

The new, combined, sentiment object, ordered by "date" and "id".

Author(s)

Samuel Borms

Examples

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

l1 <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")])
l2 <- sento_lexicons(list_lexicons[c("FEEL_en_tr")])

corp1 <- sento_corpus(corpusdf = usnews[1:200, ])
corp2 <- sento_corpus(corpusdf = usnews[201:450, ])
corp3 <- sento_corpus(corpusdf = usnews[401:700, ])

s1 <- compute_sentiment(corp1, l1, "proportionalPol")
s2 <- compute_sentiment(corp2, l1, "counts")
s3 <- compute_sentiment(corp3, l1, "counts")
s4 <- compute_sentiment(corp2, l1, "counts", do.sentence = TRUE)
s5 <- compute_sentiment(corp3, l2, "proportional", do.sentence = TRUE)
s6 <- compute_sentiment(corp3, l1, "counts", do.sentence = TRUE)

# straightforward row-wise merge
sb1 <- merge(s1, s2, s3)
nrow(sb1) # 700

# another straightforward row-wise merge
sb2 <- merge(s4, s6)

# merge of sentence and non-sentence calculations
sb3 <- merge(s3, s6)

# different methods add rows and/or columns
sb4 <- merge(s4, s5)
nrow(sb4) > nrow(sb2) # TRUE
```

nmeasures *Get number of sentiment measures*

Description

Returns the number of sentiment measures.

Usage

```
nmeasures(sento_measures)
```

Arguments

sento_measures	a sento_measures object created using sento_measures.

Value

The number of sentiment measures in the input sento_measures object.

Author(s)

Samuel Borms

nobs.sento_measures *Get number of observations in the sentiment measures*

Description

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

Usage

```
## S3 method for class 'sento_measures'
nobs(object, ...)
```

Arguments

object	a sento_measures object created using sento_measures.
...	not used.

Value

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

Author(s)

Samuel Borms

peakdates *Extract dates related to sentiment time series peaks*

Description

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

Usage

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

Arguments

sento_measures	a sento_measures object created using sento_measures.
n	a positive numeric value to indicate the number of dates associated to sentiment peaks to extract. If n < 1, it is interpreted as a quantile (for example, 0.07 would mean the 7% most extreme dates).
type	a character value, either "pos", "neg" or "both", respectively to look for the n dates related to the most positive, most negative or most extreme (in absolute terms) sentiment occurrences.
do.average	a logical to indicate whether peaks should be selected based on the average sentiment value per date.

Value

A vector of type "Date" corresponding to the n extracted sentiment peak dates.

Author(s)

Samuel Borms

Examples

```
set.seed(505)

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

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

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

---

peakdocs *Extract documents related to sentiment peaks*

Description

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

Usage

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

Arguments

sentiment	a sentiment object created using compute_sentiment or to_sentiment.
n	a positive numeric value to indicate the number of dates associated to sentiment peaks to extract. If n < 1, it is interpreted as a quantile (for example, 0.07 would mean the 7% most extreme dates).
type	a character value, either "pos", "neg" or "both", respectively to look for the n dates related to the most positive, most negative or most extreme (in absolute terms) sentiment occurrences.
do.average	a logical to indicate whether peaks should be selected based on the average sentiment value per date.

Value

A vector of type "character" corresponding to the n extracted document identifiers.

Author(s)

Samuel Borms

Examples

```
set.seed(505)

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

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

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

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

`plot.attributions` *Plot prediction attributions at specified level*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Samuel Borms, Keven Bluteau

plot.sento_measures *Plot sentiment measures*

Description

Plotting method that shows all sentiment measures from the provided sento_measures object in one plot, or the average along one of the lexicons, features and time weighting dimensions. We suggest to make use of a measures_xyz function when you want to plot only a subset of the sentiment measures.

Usage

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

Arguments

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

Value

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

Author(s)

Samuel Borms

Examples

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

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

# plot sentiment measures
```

```
plot(sento_measures, group = "features")

## Not run:
# adjust appearance of plot
library("ggplot2")
p <- plot(sento_measures)
p <- p +
  scale_x_date(name = "month-year") +
  scale_y_continuous(name = "newName")
p
## End(Not run)
```

`plot.sento_modelIter` *Plot iterative predictions versus realized values*

Description

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

Usage

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

Arguments

`x`	a `sento_modelIter` object created using `sento_model`.
`...`	not used.

Details

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

Value

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

Author(s)

Samuel Borms

predict.sento_model *Make predictions from a sento_model object*

Description

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

Usage

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

Arguments

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

Value

A prediction output depending on the type argument.

Author(s)

Samuel Borms

See Also

predict.glmnet, sento_model

scale.sento_measures *Scaling and centering of sentiment measures*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Samuel Borms

Examples

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

set.seed(505)

# construct a sento_measures object to start with
```

```
corpus <- sento_corpus(corpusdf = usnews)
corpusSample <- quanteda::corpus_sample(corpus, size = 500)
l <- sento_lexicons(list_lexicons[c("LM_en", "HENRY_en")], list_valence_shifters[["en"]])
ctr <- ctr_agg(howTime = c("equal_weight", "linear"), by = "year", lag = 3)
sento_measures <- sento_measures(corpusSample, l, ctr)

# scale sentiment measures to zero mean and unit standard deviation
sc1 <- scale(sento_measures)

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

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

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

`sento_app` *Shiny application for sentiment computation and visualisation*

Description

A Shiny application to showcase many of the package's functionalities.

Usage

```
sento_app()
```

Details

This Shiny application demonstrates mainly the `compute_sentiment`, `corpus_summarize` and `aggregate.sentiment` functions to gain insights in an uploaded corpus. The corpus should be uploaded in .csv format (by default the `usnews` dataset is displayed). Lexicons and valence shifters can be chosen from the built-in options or uploaded, and the weighting schemes are those available in the package. All calculated values and statistics can be downloaded as a .csv file also.

Author(s)

Jeroen Van Pelt, Samuel Borms

sento_corpus *Create a sento_corpus object*

Description

Formalizes a collection of texts into a sento_corpus object derived from the **quanteda** corpus object. The **quanteda** package provides a robust text mining infrastructure (see quanteda), including a handy corpus manipulation toolset. This function performs a set of checks on the input data and prepares the corpus for further analysis by structurally integrating a date dimension and numeric metadata features.

Usage

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

Arguments

corpusdf	a data.frame (or a data.table, or a tbl) with as named columns: a document "id" column (in character mode), a "date" column (as "yyyy-mm-dd"), a "texts" column (in character mode), an optional "language" column (in character mode), and a series of feature columns of type numeric, with values between 0 and 1 to specify the degree of connectedness of a feature to a document. Features could be for instance topics (e.g., legal or economic) or article sources (e.g., online or print). When no feature column is provided, a feature named "dummyFeature" is added. All spaces in the names of the features are replaced by '_'. Feature columns with values not between 0 and 1 are rescaled column-wise.
do.clean	a logical, if TRUE all texts undergo a cleaning routine to eliminate common textual garbage. This includes a brute force replacement of HTML tags and non-alphanumeric characters by an empty string. To use with care if the text is meant to have non-alphanumeric characters! Preferably, cleaning is done outside of this function call.

Details

A sento_corpus object is a specialized instance of a **quanteda** corpus. Any **quanteda** function applicable to its corpus object can also be applied to a sento_corpus object. However, changing a given sento_corpus object too drastically using some of **quanteda**'s functions might alter the very structure the corpus is meant to have (as defined in the corpusdf argument) to be able to be used as an input in other functions of the **sentometrics** package. There are functions, including corpus_sample or corpus_subset, that do not change the actual corpus structure and may come in handy. To add additional features, use add_features. Binary features are useful as a mechanism to select the texts which have to be integrated in the respective feature-based sentiment measure(s), but applies only when do.ignoreZeros = TRUE. Because of this (implicit) selection that can be performed, having complementary features (e.g., "economy" and "noneconomy") makes sense.

It is also possible to add one non-numerical feature, that is, "language", to designate the language of the corpus texts. When this feature is provided on corpus-level, a list of lexicons for different languages is expected in the function compute_sentiment.

Value

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

Author(s)

Samuel Borms

See Also

corpus, add_features

Examples

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

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

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

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

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

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

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

# corpus creation with a qualitative language feature
usnews[["language"]] <- "en"
usnews[["language"]][c(200:400)] <- "nl"
corpusLang <- sento_corpus(corpusdf = usnews)
```

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

Description

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

Usage

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

Arguments

`lexiconsIn`	a named `list` of (raw) lexicons, each element as a `data.table` or a `data.frame` with respectively a `character` column (the words) and a `numeric` column (the polarity scores). This argument can be one of the built-in lexicons accessible via `list_lexicons`.
`valenceIn`	a single valence word list as a `data.table` or a `data.frame` with respectively a `"x"` and a `"y"` or `"t"` column. The first column has the words, `"y"` has the values for bigram shifting, and `"t"` has the types of the valence shifter for a clustered approach to sentiment calculation (supported types: `1` = negators, `2` = amplifiers, `3` = deamplifiers, `4` = adversative conjunctions). If three columns are provided, the first two will be considered only. This argument can be one of the built-in valence word lists accessible via `list_valence_shifters`. A word that appears in both a lexicon and the valence word list is prioritized as a lexical entry during sentiment calculation. If `NULL`, valence shifting is not applied in the sentiment analysis.
`do.split`	a `logical` that if `TRUE` splits every lexicon into a separate positive polarity and negative polarity lexicon.

Value

A `list` of class `sento_lexicons` with each lexicon as a separate element according to its name, as a `data.table`, and optionally an element named `valence` that comprises the valence words. Every `"x"` column contains the words, every `"y"` column contains the polarity scores. The `"t"` column for valence shifters contains the different types.

Author(s)

Samuel Borms

Examples

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

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

# including a self-made lexicon, with and without valence shifters
lexIn <- c(list(myLexicon = data.table(w = c("nice", "boring"), s = c(2, -1))),
           list_lexicons[c("GI_en")])
valIn <- list_valence_shifters[["en"]]
l2 <- sento_lexicons(lexIn)
l3 <- sento_lexicons(lexIn, valIn)
l4 <- sento_lexicons(lexIn, valIn[, c("x", "y")], do.split = TRUE)
l5 <- sento_lexicons(lexIn, valIn[, c("x", "t")], do.split = TRUE)
l6 <- l5[c("GI_en_POS", "valence")] # preserves sento_lexicons class

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

## Not run:
# faulty extraction, no replacement allowed
l5["valence"]
l2[0]
l3[22]
l4[1] <- l2[1]
l4[[1]] <- l2[[1]]
l4$GI_en_NEG <- l2$myLexicon
## End(Not run)
```

`sento_measures` *One-way road towards a sento_measures object*

Description

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

Usage

```
sento_measures(sento_corpus, lexicons, ctr)
```

Arguments

`sento_corpus`	a `sento_corpus` object created with `sento_corpus`.

lexicons	a sentolexicons object created with sento_lexicons.
ctr	output from a ctr_agg call.

Details

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

Value

A sento_measures 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.
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 document-level sentiment scores data.table with "date", "word_count" and lexicon-feature sentiment scores columns. The "date" column has the dates converted at the frequency for across-document aggregation. All zeros are replaced by NA if ctr$docs$weightingParam$do.ignoreZeros = TRUE.
attribWeights	a list of document and time weights used in the attributions function. Serves further no direct purpose.
ctr	a list encapsulating the control parameters.

Author(s)

Samuel Borms, Keven Bluteau

See Also

compute_sentiment, aggregate, measures_update

Examples

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

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

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

`sento_model` *Optimized and automated sentiment-based sparse regression*

Description

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

Usage

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

Arguments

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

Details

Models are computed using the elastic net regularization as implemented in the **glmnet** package, to account for the multidimensionality of the sentiment measures. Independent variables are normalized in the regression process, but coefficients are returned in their original space. For a helpful introduction to **glmnet**, we refer to their vignette. The optimal elastic net parameters `lambda` and `alpha` are calibrated either through a to specify information criterion or through cross-validation (based on the "rolling forecasting origin" principle, using the `train` function). In the latter case, the training metric is automatically set to `"RMSE"` for a linear model and to `"Accuracy"` for a logistic model. We suppress many of the details that can be supplied to the `glmnet` and `train` functions we rely on, for the sake of user-friendliness.

Value

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

reg	optimized regression, i.e., a model-specific glmnet object, including for example the estimated coefficients.
model	the input argument ctr$model, to indicate the type of model estimated.
alpha	calibrated alpha.
lambda	calibrated lambda.
trained	output from train call (if ctr$type = "cv"). There is no such output if the control parameters alphas and lambdas both specify one value.
ic	a list composed of two elements: under "criterion", the type of information criterion used in the calibration, and under "matrix", a matrix of all information criterion values for alphas as rows and the respective lambda values as columns (if ctr$type != "cv"). Any NA value in the latter element means the specific information criterion could not be computed.
dates	sample reference dates as a two-element character vector, being the earliest and most recent date from the sento_measures object accounted for in the estimation window.
nVar	a vector of size two, with respectively the number of sentiment measures, and the number of other explanatory variables inputted.
discarded	a named logical vector of length equal to the number of sentiment measures, in which TRUE indicates that the particular sentiment measure has not been considered in the regression process. A sentiment measure is not considered when it is a duplicate of another, or when at least 50% of the observations are equal to zero.

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

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

Author(s)

Samuel Borms, Keven Bluteau

See Also

ctr_model, glmnet, train, attributions

Examples

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

set.seed(505)

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

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

# a linear model based on the Akaike information criterion
ctrIC <- ctr_model(model = "gaussian", type = "AIC", do.iter = FALSE, h = 4,
                   do.difference = TRUE)
out1 <- sento_model(sento_measures, y, x = x, ctr = ctrIC)

# attribution and prediction as post-analysis
attributions1 <- attributions(out1, sento_measures,
                              refDates = get_dates(sento_measures)[20:25])
plot(attributions1, "features")

nx <- nmeasures(sento_measures) + ncol(x)
newx <- runif(nx) * cbind(as.data.table(sento_measures)[, -1], x)[30:40, ]
preds <- predict(out1, newx = as.matrix(newx), type = "link")

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

```
# plot predicted vs. realized values
p <- plot(out2)
p

# a cross-validation based model, parallelized
cl <- parallel::makeCluster(2)
doParallel::registerDoParallel(cl)
ctrCV <- ctr_model(model = "gaussian", type = "cv", do.iter = FALSE,
                   h = 0, alphas = c(0.10, 0.50, 0.90), trainWindow = 70,
                   testWindow = 10, oos = 0, do.progress = TRUE)
out3 <- sento_model(sento_measures, y, x = x, ctr = ctrCV)
parallel::stopCluster(cl)
foreach::registerDoSEQ()
summary(out3)

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

`subset.sento_measures` *Subset sentiment measures*

Description

Subsets rows of the sentiment measures based on its columns.

Usage

```
## S3 method for class 'sento_measures'
subset(x, subset = NULL, select = NULL,
  delete = NULL, ...)
```

Arguments

`x`	a `sento_measures` object created using `sento_measures`.
`subset`	a logical (non-`character`) expression indicating the rows to keep. If a `numeric` input is given, it is used for row index subsetting.
`select`	a `character` vector of the lexicon, feature and time weighting scheme names, to indicate which measures need to be selected, or as a `list` of `character` vectors, possibly with separately specified combinations (consisting of one unique lexicon, one unique feature, and one unique time weighting scheme at maximum).
`delete`	see the `select` argument.
`...`	not used.

Value

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

Author(s)

Samuel Borms

Examples

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

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

# three specified indices in required list format
three <- as.list(stringi::stri_split(
  c("LM_en--economy--linear",
    "HENRY_en--wsj--equal_weight",
    "HENRY_en--wapo--equal_weight"), regex = "--")
)

# different subsets
sub1 <- subset(sm, HENRY_en--economy--equal_weight >= 0.01)
sub2 <- subset(sm, date %in% get_dates(sm)[3:12])
sub3 <- subset(sm, 3:12)
sub4 <- subset(sm, 1:100) # warning

# different selections
sel1 <- subset(sm, select = "equal_weight")
sel2 <- subset(sm, select = c("equal_weight", "linear"))
sel3 <- subset(sm, select = c("linear", "LM_en"))
sel4 <- subset(sm, select = list(c("linear", "wsj"), c("linear", "economy")))
sel5 <- subset(sm, select = three)

# different deletions
del1 <- subset(sm, delete = "equal_weight")
del2 <- subset(sm, delete = c("linear", "LM_en"))
del3 <- subset(sm, delete = list(c("linear", "wsj"), c("linear", "economy")))
del4 <- subset(sm, delete = c("equal_weight", "linear")) # warning
del5 <- subset(sm, delete = three)
```

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

Description

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

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

Usage

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

Format

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

Source

Economic News Article Tone and Relevance. Retrieved November 1, 2017.

Examples

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

weights_almon *Compute Almon polynomials*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

See Also

ctr_agg

weights_beta *Compute Beta weighting curves*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

References

Ghysels, Sinko and Valkanov (2007). "MIDAS regressions: Further results and new directions". *Econometric Reviews 26, 53-90*, https://doi.org/10.1080/07474930600972467.

See Also

ctr_agg

---

weights_exponential *Compute exponential weighting curves*

Description

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

Usage

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

Arguments

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

Value

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

See Also

ctr_agg

Index