Package 'polmineR'

January 8, 2019

Type Package

Title Toolkit for Corpus Analysis

Version 0.7.11

Date 2019-01-07

Imports methods, R6, data.table, slam, Matrix, tm, DT, xml2, stringi, utils, jsonlite, parallel, pbapply, RcppCWB (>= 0.2.2), knitr

Suggests markdown, rmarkdown, htmltools, openxlsx, sendmailR, shiny, shinythemes, testthat, tidytext, magrittr, covr, igraph

VignetteBuilder knitr

LazyData yes

Description Library for corpus analysis using the Corpus Workbench as an efficient back end for indexing and querying large corpora. The package offers functionality to flexibly create partitions and to carry out basic statistical operations (count, co-occurrences etc.). The original full text of documents can be reconstructed and inspected at any time. Beyond that, the package is intended to serve as an interface to packages implementing advanced statistical procedures. Respective data structures (document term matrices, term co-occurrence matrices etc.) can be created based on the indexed corpora.

BugReports https://github.com/PolMine/polmineR/issues

Biarch true

License GPL-3

URL https://www.github.com/PolMine/polmineR

Encoding UTF-8

Collate 'CQI.R' 'Labels.R' 'polmineR.R' 'S4classes.R' 'p_attributes.R' 'textstat.R' 'bundle.R' 'corpus.R' 'count.R' 'partition.R' 'partition_bundle.R' 'ngrams.R' 'features.R' 'context.R' 'TermDocumentMatrix.R' 'as.VCorpus.R' 'as.markdown.R' 'kwic.R' 'decode.R' 'cooccurrences.R' 'as.sparseMatrix.R' 'as.speeches.R' 'blapply.R' 'browse.R' 'hits.R' 'tempcorpus.R' 'cpos.R' 'cqpserver.R' 'dispersion.R' 'dotplot.R' 'encoding.R' 'enrich.R' 'format.R' 'highlight.R' 'html.R' 'label.R' 'mail.R'

'means.R' 'noise.R' 'regions.R' 'read.R' 'registry.R'
'reindex.R' 'renamed.R' 's_attributes.R' 'size.R' 'stats.R'
'store.R' 't_test.R' 'templates.R' 'terms.R' 'token_stream.R'
'tooltips.R' 'trim.R' 'type.R' 'use.R' 'utils.R' 'view.R'
'weigh.R' 'zzz.R'

RoxygenNote 6.1.1

NeedsCompilation no

Author Andreas Blaette [aut, cre] (<https://orcid.org/0000-0001-8970-8010>)

Maintainer Andreas Blaette <andreas.blaette@uni-due.de>

Repository CRAN

Date/Publication 2019-01-08 18:00:07 UTC

R topics documented:

polmineR-package *polmineR-package*

Description

A library for corpus analysis using the Corpus Workbench (CWB) as an efficient back end for indexing and querying large corpora.

Usage

```
polmineR()
```

Details

The package offers functionality to flexibly create partitions and to carry out basic statistical operations (count, co-occurrences etc.). The original full text of documents can be reconstructed and inspected at any time. Beyond that, the package is intended to serve as an interface to packages implementing advanced statistical procedures. Respective data structures (document term matrices, term co- occurrence matrices etc.) can be created based on the indexed corpora.

A session registry directory (see `registry()`) combines the registry files for corpora that may reside in anywhere on the system. Upon loading `polmineR`, the files in the registry directory defined by the environment variable CORPUS_REGISTRY are copied to the session registry directory. To see whether the environment variable CORPUS_REGISTRY is set, use the 'Sys.getenv()'-function. Corpora wrapped in R data packages can be activated using the function `use()`.

The package includes a draft shiny app that can be called using `polmineR()`.

Author(s)

Andreas Blaette (andreas.blaette@uni-due.de)

References

Jockers, Matthew L. (2014): *Text Analysis with R for Students of Literature*. Cham et al: Springer.

Baker, Paul (2006): *Using Corpora in Discourse Analysis*. London: continuum.

Examples

```
use("polmineR") # activate demo corpora included in the package

# Core methods applied to corpus

C <- count("REUTERS", query = "oil")
C <- count("REUTERS", query = c("oil", "barrel"))
C <- count("REUTERS", query = '"Saudi" "Arab.*"', breakdown = TRUE, cqp = TRUE)
D <- dispersion("REUTERS", query = "oil", s_attribute = "id")
K <- kwic("REUTERS", query = "oil")
CO <- cooccurrences("REUTERS", query = "oil")
```

```
# Core methods applied to partition

kuwait <- partition("REUTERS", places = "kuwait", regex = TRUE)
C <- count(kuwait, query = "oil")
D <- dispersion(kuwait, query = "oil", s_attribute = "id")
K <- kwic(kuwait, query = "oil", meta = "id")
CO <- cooccurrences(kuwait, query = "oil")


# Go back to full text

p <- partition("REUTERS", id = 127)
read(p)
h <- html(p)
h_highlighted <- highlight(h, highlight = list(yellow = "oil"))
h_highlighted


# Generate term document matrix

pb <- partition_bundle("REUTERS", s_attribute = "id")
cnt <- count(pb, p_attribute = "word")
tdm <- as.TermDocumentMatrix(cnt, col = "count")
```

---

as.markdown *Get markdown-formatted full text of a partition.*

Description

The method is the worker behind the read-method, which will be called usually to reconstruct the full text of a partition and read it. The as.markdown-method can be customized for different classes inheriting from the partition-class.

Usage

```
as.markdown(.Object, ...)

## S4 method for signature 'partition'
as.markdown(.Object,
  meta = getOption("polmineR.meta"), template = get_template(.Object),
  cpos = TRUE, cutoff = NULL, verbose = FALSE, ...)

## S4 method for signature 'plpr_partition'
as.markdown(.Object, meta = NULL,
  template = get_template(.Object), cpos = FALSE,
  interjections = TRUE, cutoff = NULL, ...)
```

Arguments

.Object	The object to be converted, a partition, or a class inheriting from partition, such as plpr_partition.
...	further arguments
meta	The metainformation (s-attributes) to be displayed.
template	A template for formating output.
cpos	A logical value, whether to add cpos as ids in span elements.
cutoff	The maximum number of tokens to reconstruct, to avoid that full text is excessively long.
verbose	A logical value, whether to output messages.
interjections	A logical value, whether to format interjections.

Examples

```
use("polmineR")
P <- partition("REUTERS", places = "argentina")
as.markdown(P)
as.markdown(P, meta = c("id", "places"))
if (interactive()) read(P, meta = c("id", "places"))
```

---

as.sparseMatrix *Type conversion - get sparseMatrix.*

Description

Turn objects into the sparseMatrix as defined in the Matrix package.

Usage

```
as.sparseMatrix(x, ...)

## S4 method for signature 'simple_triplet_matrix'
as.sparseMatrix(x, ...)

## S4 method for signature 'TermDocumentMatrix'
as.sparseMatrix(x, ...)

## S4 method for signature 'bundle'
as.sparseMatrix(x, col, ...)
```

Arguments

x	object to convert
...	Further arguments that are passed to a call to sparseMatrix. Can be used, for instance to set giveCsparse to FALSE to get a dgTMatrix, not a dgCMatrix.
col	column name to get values from (if x is a bundle)

as.speeches *Split corpus or partition into speeches.*

Description

Split entire corpus or a partition into speeches. The heuristic is to split the corpus/partition into partitions on day-to-day basis first, using the s-attribute provided by s_attribute_date. These subcorpora are then splitted into speeches by speaker name, using s-attribute s_attribute_name. If there is a gap larger than the number of tokens supplied by argument gap, contributions of a speaker are assumed to be two seperate speeches.

Usage

```
as.speeches(.Object, s_attribute_date = grep("date",
  s_attributes(.Object), value = TRUE), s_attribute_name = grep("name",
  s_attributes(.Object), value = TRUE), gap = 500, mc = FALSE,
  verbose = TRUE, progress = TRUE)
```

Arguments

.Object	A partition, or length-one character vector indicating a CWB corpus.
s_attribute_date	The s-attribute that provides the dates of sessions.
s_attribute_name	The s-attribute that provides the names of speakers.
gap	Number of tokens between strucs assumed to make the difference whether a speech has been interrupted (by an interjection or question), or whether to assume seperate speeches.
mc	Whether to use multicore, defaults to FALSE.
verbose	A logical value, defaults to TRUE.
progress	logical

Value

A partition_bundle, the names of the objects in the bundle are the speaker name, the date of the speech and an index for the number of the speech on a given day, concatenated by underscores.

Examples

```
use("polmineR")
speeches <- as.speeches(
  "GERMAPARLMINI",
  s_attribute_date = "date", s_attribute_name = "speaker"
)
speeches_count <- count(speeches, p_attribute = "word")
tdm <- as.TermDocumentMatrix(speeches_count, col = "count")
```

```
bt <- partition("GERMAPARLMINI", date = "2009-10-27")
speeches <- as.speeches(bt, s_attribute_name = "speaker")
summary(speeches)
```

`as.TermDocumentMatrix` *Generate TermDocumentMatrix / DocumentTermMatrix.*

Description

Methods to generate the classes `TermDocumentMatrix` or `DocumentTermMatrix` as defined in the `tm` package. These classes inherit from the `simple_triplet_matrix`-class defined in the `slam`-package. There are many text mining applications for document-term matrices. A `DocumentTermMatrix` is required as input by the `topicmodels` package, for instance.

Usage

```
as.TermDocumentMatrix(x, ...)

as.DocumentTermMatrix(x, ...)

## S4 method for signature 'character'
as.TermDocumentMatrix(x, p_attribute, s_attribute,
  verbose = TRUE, ...)

## S4 method for signature 'character'
as.DocumentTermMatrix(x, p_attribute, s_attribute,
  verbose = TRUE, ...)

## S4 method for signature 'bundle'
as.TermDocumentMatrix(x, col, p_attribute = NULL,
  verbose = TRUE, ...)

## S4 method for signature 'bundle'
as.DocumentTermMatrix(x, col, p_attribute = NULL,
  verbose = TRUE, ...)

## S4 method for signature 'partition_bundle'
as.TermDocumentMatrix(x, p_attribute = NULL,
  col = NULL, verbose = TRUE, ...)

## S4 method for signature 'partition_bundle'
as.DocumentTermMatrix(x, p_attribute = NULL,
  col = NULL, verbose = TRUE, ...)

## S4 method for signature 'context'
as.DocumentTermMatrix(x, p_attribute, verbose = TRUE,
```

```
  ...)

## S4 method for signature 'context'
as.TermDocumentMatrix(x, p_attribute, verbose = TRUE,
  ...)
```

Arguments

x	a character vector indicating a corpus, or an object of class bundle, or inheriting from class bundle (e.g. partition_bundle)
...	s-attribute definitions used for subsetting the corpus, compare partition-method
p_attribute	p-attribute counting is be based on
s_attribute	s-attribute that defines content of columns, or rows
verbose	logial, whether to output progress messages
col	the column of data.table in slot stat (if x is a bundle) to use of assembling the matrix

Details

The method can be applied on objects of the class character, bundle, or classes inheriting from the bundle class.

If x refers to a corpus (i.e. is a length 1 character vector), a TermDocumentMatrix, or DocumentTermMatrix will be generated for subsets of the corpus based on the s_attribute provided. Counts are performed for the p_attribute. Further parameters provided (passed in as ... are interpreted as s-attributes that define a subset of the corpus for splitting it according to s_attribute. If struc values for s_attribute are not unique, the necessary aggregation is performed, slowing things somewhat down.

If x is a bundle or a class inheriting from it, the counts or whatever measure is present in the stat slots (in the column indicated by col) will be turned into the values of the sparse matrix that is generated. A special case is the generation of the sparse matrix based on a partition_bundle that does not yet include counts. In this case, a p_attribute needs to be provided. Then counting will be performed, too.

Value

a TermDocumentMatrix

Author(s)

Andreas Blaette

Examples

```
use("polmineR")

# do-it-yourself
p <- partition("GERMAPARLMINI", date = ".*", regex = TRUE)
pB <- partition_bundle(p, s_attribute = "date")
```

```
pB <- enrich(pB, p_attribute="word")
tdm <- as.TermDocumentMatrix(pB, col = "count")

 # leave the counting to the as.TermDocumentMatrix-method
pB2 <- partition_bundle(p, s_attribute = "date")
tdm <- as.TermDocumentMatrix(pB2, p_attribute = "word", verbose = TRUE)

# diretissima
tdm <- as.TermDocumentMatrix("GERMAPARLMINI", p_attribute = "word", s_attribute = "date")
```

---

as.VCorpus *Coerce partition_bundle to VCorpus.*

Description

Retrieve full text for the partition objects in a partition_bundle and generate a VCorpus-class object from the tm-package.

Usage

```
## S4 method for signature 'partition_bundle'
as.VCorpus(x)
```

Arguments

x	A partition_bundle object.

Details

The VCorpus class of the tm-package offers an interface to access the functionality of the tm-package. Note however that generating a VCorpus to get a DocumentTermMatrix, or a TermDocumentMatrix is a highly inefficient detour. Applying the as.DocumentTermMatrix or as.TermDocumentMatrix methods on a partition_bundle is the recommended approach.

If the tm-package has been loaded, the as.VCorpus-method included in the polmineR-package may become inaccessible. To deal with this (propable) scenario, it is possible to use a coerce-method (as(YOUROBJECT, "VCorpus")), see examples.

Examples

```
use("polmineR")
p <- partition("GERMAPARLMINI", date = "2009-11-10")
pb <- partition_bundle(p, s_attribute = "speaker")
vc <- as.VCorpus(pb) # works only, if tm-package has not yet been loaded
vc <- as(pb, "VCorpus") # will work if tm-package has been loaded, too
```

blapply *apply a function over a list or bundle*

Description

Very similar to lapply, but applicable to bundle-objects, in particular. The purpose of the method is to supply a uniform und convenient parallel backend for the polmineR package. In particular, progress bars are supported (the naming of the method is derived from bla bla).

Usage

```
blapply(x, ...)

## S4 method for signature 'list'
blapply(x, f, mc = TRUE, progress = TRUE,
  verbose = FALSE, ...)

## S4 method for signature 'vector'
blapply(x, f, mc = FALSE, progress = TRUE,
  verbose = FALSE, ...)

## S4 method for signature 'bundle'
blapply(x, f, mc = FALSE, progress = TRUE,
  verbose = FALSE, ...)
```

Arguments

x	a list or a bundle object
...	further parameters
f	a function that can be applied to each object contained in the bundle, note that it should swallow the parameters mc, verbose and progress (use ... to catch these params)
mc	logical, whether to use multicore - if TRUE, the number of cores will be taken from the polmineR-options
progress	logical, whether to display progress bar
verbose	logical, whether to print intermediate messages

Examples

```
use("polmineR")
bt <- partition("GERMAPARLMINI", date = ".*", regex=TRUE)
speeches <- as.speeches(bt, s_attribute_date = "date", s_attribute_name = "speaker")
foo <- blapply(speeches, function(x, ...) slot(x, "cpos"))
```

browse *Display in browser*

Description

Display in browser

Usage

```
browse(object, ...)

## S4 method for signature 'textstat'
browse(object)

## S4 method for signature 'cooccurrences'
browse(object)

## S4 method for signature 'partition'
browse(object, meta = NULL)

## S4 method for signature 'html'
browse(object)

## S4 method for signature 'kwic'
browse(object, colnames = NULL)

## S4 method for signature 'press_partition'
browse(object, meta = c("text_newspaper",
  "text_date"))
```

Arguments

object	what is to be displayed
...	further parameters
meta	metainformation to be displayed
colnames	colnames to be used for data.frame

bundle-class *Bundle Class*

Description

A bundle is used to combine several objects (partition, context, features, cooccurrences objects) into one S4 class object. Typically, a class inheriting from the bundle superclass will be used. When working with a context_bundle, a features_bundle, a cooccurrences_bundle, or a context_bundle, a similar set of standard methods is available to perform transformations.

Usage

```
## S4 replacement method for signature 'bundle,character'
name(x) <- value

## S4 method for signature 'bundle'
length(x)

## S4 method for signature 'bundle'
names(x)

## S4 replacement method for signature 'bundle,vector'
names(x) <- value

## S4 method for signature 'bundle'
unique(x)

## S4 method for signature 'bundle,bundle'
e1 + e2

## S4 method for signature 'bundle,textstat'
e1 + e2

## S4 method for signature 'bundle'
x[[i]]

## S4 replacement method for signature 'bundle'
x[[i]] <- value

## S4 method for signature 'bundle'
x$name

## S4 replacement method for signature 'bundle'
x$name <- value

## S4 method for signature 'bundle'
sample(x, size)

## S4 method for signature 'list'
as.bundle(object, ...)

## S4 method for signature 'textstat'
as.bundle(object)

## S3 method for class 'bundle'
as.data.table(x, col)

## S4 method for signature 'bundle'
as.matrix(x, col)
```

```
## S4 method for signature 'bundle'
subset(x, ...)

## S4 method for signature 'bundle'
as.list(x)
```

Arguments

x	a bundle object
value	character string with a name to be assigned
e1	object 1
e2	object 2
i	integer to index a bundle object
name	The name of an object in the bundle object.
size	number of items to choose to generate a sample
object	a bundle object
...	further parameters
col	columns of the data.table to use to generate an object

Slots

corpus The CWB corpus the objects in the bundle are based on, a length 1 character vector.

objects An object of class "list"

p_attribute Object of class "character"

encoding The encoding of the corpus.

Author(s)

Andreas Blaette

Examples

```
parties <- s_attributes("GERMAPARLMINI", "party")
parties <- parties[-which(parties == "NA")]
party_bundle <- partition_bundle("GERMAPARLMINI", s_attribute = "party")
length(party_bundle)
names(party_bundle)
party_bundle <- enrich(party_bundle, p_attribute = "word")
summary(party_bundle)
parties_big <- party_bundle[[c("CDU_CSU", "SPD")]]
summary(parties_big)
p <- partition("GERMAPARLMINI", date = "2009-11-11")
pb <- partition_bundle(p, s_attribute = "party")
names(pb)
pb[["NA"]] <- NULL
names(pb)
```

```
pb <- partition_bundle("GERMAPARLMINI", s_attribute = "party")
pb$SPD # access partition names "SPD" in partition_bundle pb
pb <- partition_bundle("GERMAPARLMINI", s_attribute = "party")
pb$"NA" <- NULL # quotation needed if name is "NA"
use("polmineR")
Ps <- partition_bundle(
  "REUTERS", s_attribute = "id",
  values = s_attributes("REUTERS", "id")
)
Cs <- cooccurrences(Ps, query = "oil", cqp = FALSE, verbose = FALSE, progress = TRUE)
dt <- polmineR:::as.data.table.bundle(Cs, col = "ll")
m <- as.matrix(Cs, col = "ll")
```

`chisquare` *Perform chisquare-text.*

Description

Perform Chisquare-Test based on a table with counts

Usage

```
chisquare(.Object)

## S4 method for signature 'features'
chisquare(.Object)

## S4 method for signature 'context'
chisquare(.Object)

## S4 method for signature 'cooccurrences'
chisquare(.Object)
```

Arguments

`.Object`	A `features` object, or an object inheriting from it (`context`, `cooccurrences`).

Details

The basis for computing for the chi square test is a contingency table of observationes, which is prepared for every single token in the corpus. It reports counts for a token to inspect and all other tokens in a corpus of interest (coi) and a reference corpus (ref):

	coi	ref	TOTAL
count token	o_{11}	o_{12}	r_1
other tokens	o_{21}	o_{22}	r_2
TOTAL	c_1	c_2	N

Based on the contingency table, expected values are calculated for each cell, as the product of the column and margin sums, divided by the overall number of tokens (see example). The standard formula for calculating the chi-square test is computed as follows.

$$X^2 = \sum \frac{(O_{ij} - E_{ij})^2}{O_{ij}}$$

Results from the chisquare test are only robust for at least 5 observed counts in the corpus of interest. Usually, results need to be filtered accordingly (see examples).

Value

Same class as input object, with enriched table in the `stat`-slot.

Author(s)

Andreas Blaette

References

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 169-172.

Kilgarriff, A. and Rose, T. (1998): Measures for corpus similarity and homogeneity. *Proc. 3rd Conf. on Empirical Methods in Natural Language Processing*. Granada, Spain, pp 46-52.

See Also

See `ll`, `pmi` and `t_test`.

Examples

```
use("polmineR")
library(data.table)
m <- partition(
  "GERMAPARLMINI", speaker = "Merkel", interjection = "speech",
  regex = TRUE, p_attribute = "word"
)
f <- features(m, "GERMAPARLMINI", included = TRUE)
f_min <- subset(f, count_coi >= 5)
summary(f_min)

## Not run:

# A sample do-it-yourself calculation for chisquare:

# (a) prepare matrix with observed values
o <- matrix(data = rep(NA, 4), ncol = 2)
o[1,1] <- as.data.table(m)[word == "Weg"][["count"]]
o[1,2] <- count("GERMAPARLMINI", query = "Weg")[["count"]] - o[1,1]
o[2,1] <- size(f)[["coi"]] - o[1,1]
o[2,2] <- size(f)[["ref"]] - o[1,2]
```

```
# prepare matrix with expected values, calculate margin sums first

r <- rowSums(o)
c <- colSums(o)
N <- sum(o)

e <- matrix(data = rep(NA, 4), ncol = 2)
e[1,1] <- r[1] * (c[1] / N)
e[1,2] <- r[1] * (c[2] / N)
e[2,1] <- r[2] * (c[1] / N)
e[2,2] <- r[2] * (c[2] / N)


# compute chisquare statistic

y <- matrix(rep(NA, 4), ncol = 2)
for (i in 1:2) for (j in 1:2) y[i,j] <- (o[i,j] - e[i,j])^2 / e[i,j]
chisquare_value <- sum(y)

as(f, "data.table")[word == "Weg"][["chisquare"]]

## End(Not run)
```

---

context *Analyze context of a node word.*

Description

Retrieve the word context of a token, optionally checking for boundaries of a XML region.

Usage

```
context(.Object, ...)

## S4 method for signature 'partition'
context(.Object, query, cqp = is.cqp,
  check = TRUE, left = getOption("polmineR.left"),
  right = getOption("polmineR.right"),
  p_attribute = getOption("polmineR.p_attribute"), boundary = NULL,
  stoplist = NULL, positivelist = NULL, regex = FALSE,
  count = TRUE, mc = getOption("polmineR.mc"), verbose = TRUE,
  progress = TRUE, ...)

## S4 method for signature 'character'
context(.Object, query, cqp = is.cqp,
  p_attribute = getOption("polmineR.p_attribute"), boundary = NULL,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"), stoplist = NULL,
```

```
  positivelist = NULL, regex = FALSE, count = TRUE,
  mc = getOption("polmineR.mc"), verbose = TRUE, progress = TRUE,
  ...)

## S4 method for signature 'partition_bundle'
context(.Object, query, p_attribute,
  verbose = TRUE, ...)

## S4 method for signature 'cooccurrences'
context(.Object, query, check = TRUE,
  complete = FALSE)
```

Arguments

.Object	a partition or a partition_bundle object
...	further parameters
query	A query, which may by a character vector or a CQP query.
cqp	defaults to is.cqp-function, or provide TRUE/FALSE
check	A logical value, whether to check validity of CQP query using check_cqp_query.
left	Number of tokens to the left of the query match.
right	Number of tokens to the right of the query match.
p_attribute	The p-attribute of the query.
boundary	If provided, a length-one character vector specifying a s-attribute. It will be checked that corpus positions do not extend beyond the region defined by the s-attribute.
stoplist	Exclude match for query if stopword(s) is/are are present in context. See positivelist for further explanation.
positivelist	character vector or numeric/integer vector: include a query hit only if token in positivelist is present. If positivelist is a character vector, it may include regular expressions (see parameter regex)
regex	logical, defaults to FALSE - whether stoplist and/or positivelist are regular expressions
count	logical
mc	whether to use multicore; if NULL (default), the function will get the value from the options
verbose	report progress, defaults to TRUE
progress	logical, whether to show progress bar
complete	enhance completely

Details

For formulating the query, CPQ syntax may be used (see examples). Statistical tests available are log-likelihood, t-test, pmi.

Value

depending on whether a partition or a partition_bundle serves as input, the return will be a context object, or a `context_bundle` object

Author(s)

Andreas Blaette

Examples

```
use("polmineR")
p <- partition("GERMAPARLMINI", interjection = "speech")
y <- context(p, query = "Integration", p_attribute = "word")
y <- context(p, query = "Integration", p_attribute = "word", positivelist = "Bildung")
y <- context(
  p, query = "Integration", p_attribute = "word",
  positivelist = c("[aA]rbeit.*", "Ausbildung"), regex = TRUE
)
```

---

context-class *Context class.*

Description

Class to organize information of context analysis.

Usage

```
## S4 method for signature 'context'
length(x)

## S4 method for signature 'context'
p_attributes(.Object)

## S4 method for signature 'context'
count(.Object)

## S4 method for signature 'context'
sample(x, size)

## S4 method for signature 'context'
enrich(.Object, s_attribute = NULL,
  p_attribute = NULL, decode = FALSE, verbose = TRUE, ...)

## S4 method for signature 'context'
as.regions(x, node = TRUE)

## S4 method for signature 'context'
```

```
trim(object, s_attribute = NULL,
  positivelist = NULL, p_attribute = p_attributes(object),
  regex = FALSE, stoplist = NULL, verbose = TRUE, progress = TRUE,
  ...)
```

Arguments

x	a context object
.Object	object
size	integer indicating sample size
s_attribute	s-attribute(s) to add to data.table in cpos-slot
p_attribute	p-attribute(s) to add to data.table in cpos-slot
decode	logical, whether to convert integer ids to expressive strings
verbose	logical, whether to be talkative
...	to maintain backwards compatibility if argument pAttribute is still used
node	A logical value, whether to include the node (i.e. query matches) in the region matrix generated when creating a partition from a context-object.
object	a context object
positivelist	tokens that are required to be present to keep a match
regex	logical, whether positivlist / stoplist is interpreted as regular expressions
stoplist	tokens that are used to exclude a match
progress	logical, whether to show progress bar

Details

Objects of the class context include a data.table in the slot cpos. The data.table will at least include the columns "hit_no", "cpos" and "position".

The length-method will return the number of hits that were achieved.

The enrich-method can be used to add additional information to the data.table in the "cpos"-slot of a context-object.

Slots

query The query examined (character).

count An integer value, the number of hits for the query.

partition The partition the context object is based on.

size_partition The size of the partition, a length-one integer vector.

left A length-one integer value, the number of tokens to the left of the query match.

right An integer value, the number of tokens to the right of the query match.

size A length-one integer value, the number of tokens covered by the context-object, i.e. the number of tokens in the right and left context of the node as well as query matches.

size_match A length-one integer value, the number of tokens matches by the query. Identical with the value in slot count if the query is *not* a CQP query.

`size_coi` A length-one `integer` value, the number of tokens in the right and left context of the node (excluding query matches).

`size_ref` A length-one `integer` value, the number of tokens in the partition, without tokens matched and the tokens in the left and right context.

`boundary` An s-attribute (`character`).

`p_attribute` The p-attribute of the query (`character`).

`corpus` The CWB corpus used (`character`).

`stat` A `data.table`, the statistics of the analysis.

`encoding` Object of class `character`, encoding of the corpus.

`cpos` A `data.table`, with the columns hit_no, cpos, position, word_id.

`method` A `character`-vector, statistical test used.

`call` Object of class `character`, call that generated the object.

`context_bundle-class` *S4 context_bundle class*

Description

class to organize information of multiple context analyses

Slots

`objects` Object of class `"list"` a list of context objects

Methods

show output of core information

summary core statistical information

[specific cooccurrences

[[specific cooccurrences

cooccurrences *Get cooccurrence statistics.*

Description

Get cooccurrence statistics.

Usage

```
cooccurrences(.Object, ...)

## S4 method for signature 'character'
cooccurrences(.Object, query, cqp = is.cqp,
  p_attribute = getOption("polmineR.p_attribute"), s_attribute = NULL,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"), stoplist = NULL,
  positivelist = NULL, regex = FALSE, keep = NULL, cpos = NULL,
  method = "ll", mc = getOption("polmineR.mc"), verbose = FALSE,
  progress = FALSE, ...)

## S4 method for signature 'partition'
cooccurrences(.Object, query, cqp = is.cqp,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"),
  p_attribute = getOption("polmineR.p_attribute"), s_attribute = NULL,
  stoplist = NULL, positivelist = NULL, keep = NULL, method = "ll",
  mc = FALSE, progress = TRUE, verbose = FALSE, ...)

## S4 method for signature 'context'
cooccurrences(.Object, method = "ll",
  verbose = FALSE)

## S4 method for signature 'Corpus'
cooccurrences(.Object, query,
  p_attribute = getOption("polmineR.p_attribute"), ...)

## S4 method for signature 'partition_bundle'
cooccurrences(.Object, query,
  mc = getOption("polmineR.mc"), ...)

## S4 method for signature 'Cooccurrences'
cooccurrences(.Object, query)
```

Arguments

.Object	A partition object, or a character vector with a CWB corpus.

...	Further parameters that will be passed into bigmatrix (applies only of big = TRUE).
query	A query, either a character vector to match a token, or a CQP query.
cqp	Defaults to is.cqp-function, or provide TRUE/FALSE; relevant only if query is not NULL.
p_attribute	The p-attribute of the tokens/the query.
s_attribute	If provided, it will be checked that corpus positions of windows do not extend beyond the region defined by the s-attribute.
left	Number of tokens to the left of the query match.
right	Number of tokens to the right of the query match.
stoplist	Exclude a query hit from analysis if stopword(s) is/are in context (relevant only if query is not NULL).
positivelist	Character vector or numeric vector: include a query hit only if token in positivelist is present. If positivelist is a character vector, it is assumed to provide regex expressions (incredibly long if the list is long) (relevant only if query is nut NULL)
regex	A logical value, whether stoplist/positivelist are interpreted as regular expressions.
keep	list with tokens to keep
cpos	integer vector with corpus positions, defaults to NULL - then the corpus positions for the whole corpus will be used
method	The statistical test(s) to use (defaults to "ll").
mc	whether to use multicore
verbose	A logical value, whether to be verbose.
progress	A logical value, whether to output progress bar.

Value

a cooccurrences-class object

Author(s)

Andreas Blaette

References

Baker, Paul (2006): *Using Corpora in Discourse Analysis*. London: continuum, p. 95-120 (ch. 5).

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 151-189 (ch. 5).

See Also

See the documentation for the ll-method for an explanation of the computation of the log-likelihood statistic.

Examples

```
use("polmineR")
merkel <- partition("GERMAPARLMINI", interjection = "speech", speaker = ".*Merkel", regex = TRUE)
merkel <- enrich(merkel, p_attribute = "word")
cooc <- cooccurrences(merkel, query = "Deutschland")

# use subset-method to filter results
a <- cooccurrences("REUTERS", query = "oil")
b <- subset(a, !is.na(ll))
c <- subset(b, !word %in% tm::stopwords("en"))
d <- subset(c, count_coi >= 5)
e <- subset(c, ll >= 10.83)
format(e)

# using pipe operator may be convenient
if (require(magrittr)){
cooccurrences("REUTERS", query = "oil") %>%
  subset(!is.na(ll)) %>%
  subset(!word %in% tm::stopwords("en")) %>%
  subset(count_coi >= 5) %>%
  subset(ll >= 10.83) %>%
  format()
}
```

Cooccurrences,character-method

Get all cooccurrences in corpus/partition.

Description

Obtain all cooccurrences in a corpus, or a `partition`. The result is a `Cooccurrences`-class object which includes a `data.table` with counts of cooccurrences. See the documentation entry for the `Cooccurrences`-class for methods to process `Cooccurrences`-class objects.

Usage

```
## S4 method for signature 'character'
Cooccurrences(.Object, p_attribute, left, right,
  stoplist = NULL, mc = getOption("polmineR.mc"), verbose = FALSE,
  progress = FALSE)

## S4 method for signature 'partition'
Cooccurrences(.Object, p_attribute, left, right,
  stoplist = NULL, mc = getOption("polmineR.mc"), verbose = FALSE,
  progress = FALSE)
```

Arguments

.Object	A length-one character vector indicating a corpus, or a partition object.
p_attribute	Positional attributes to evaluate.
left	A scalar integer value, size of left context.
right	A scalar integer value, size of right context.
stoplist	Tokens to exclude from the analysis.
mc	Logical value, whether to use multiple cores.
verbose	Logical value, whether to output messages.
progress	Logical value, whether to display a progress bar.

Details

The implementation uses a data.table to store information and makes heavy use of the reference logic of the data.table package, to avoid copying potentially large objects, and to be parsimonious with limited memory. The behaviour resulting from in-place changes may be uncommon, see examples.

See Also

To learn about methods available for the object that is returned, see the documentation of the Cooccurrences-class. See the cooccurrences-method (starting with a lower case c) to get the cooccurrences for the match for a query, which may also be a CQP query.

Examples

```
## Not run:
# In a first scenario, we get all cooccurrences for the REUTERS corpus,
# excluding stopwords

stopwords <- unname(unlist(
  noise(
    terms("REUTERS", p_attribute = "word"),
    stopwordsLanguage = "en"
    )
  ))
r <- Cooccurrences(
  .Object = "REUTERS", p_attribute = "word",
  left = 5L, right = 5L, stoplist = stopwords
)
ll(r) # note that the table in the stat slot is augmented in-place
decode(r) # in-place modification, again
r <- subset(r, ll > 11.83 & ab_count >= 5)
data.table::setorderv(r@stat, cols = "ll", order = -1L)
head(r, 25)

if (requireNamespace("igraph", quietly = TRUE)){
  r@partition <- enrich(r@partition, p_attribute = "word")
  g <- as_igraph(r, as.undirected = TRUE)
```

```
  plot(g)
}

# The next scenario is a cross-check that extracting cooccurrences from
# from a Cooccurrences-class object with all cooccurrences and the result
# for getting cooccurrences for a single object are identical

a <- cooccurrences(r, query = "oil")
a <- data.table::as.data.table(a)

b <- cooccurrences("REUTERS", query = "oil", left = 5, right = 5, p_attribute = "word")
b <- data.table::as.data.table(b)
b <- b[!word %in% stopwords]

all(b[["word"]][1:5] == a[["word"]][1:5]) # needs to be identical!


stopwords <- unlist(noise(
  terms("GERMAPARLMINI", p_attribute = "word"),
  stopwordsLanguage = "german"
  )
)

# We now filter cooccurrences by keeping only the statistically
# significant cooccurrens, identified by comparison with cooccurrences
# derived from a reference corpus

plpr_partition <- partition(
  "GERMAPARLMINI", date = "2009-11-10", interjection = "speech",
  p_attribute = "word"
)
plpr_cooc <- Cooccurrences(
  plpr_partition, p_attribute = "word",
  left = 3L, right = 3L,
  stoplist = stopwords,
  verbose = TRUE
)
decode(plpr_cooc)
ll(plpr_cooc)

merkel <- partition(
  "GERMAPARLMINI", speaker = "Merkel", date = "2009-11-10", interjection = "speech",
  regex = TRUE,
  p_attribute = "word"
)
merkel_cooc <- Cooccurrences(
  merkel, p_attribute = "word",
  left = 3L, right = 3L,
  stoplist = stopwords,
  verbose = TRUE
)
decode(merkel_cooc)
ll(merkel_cooc)
```

```
merkel_min <- subset(
  merkel_cooc,
  by = subset(features(merkel_cooc, plpr_cooc), rank_ll <= 50)
  )

# Esentially the same procedure as in the previous example, but with
# two positional attributes, so that part-of-speech annotation is
# used for additional filtering.


protocol <- partition(
  "GERMAPARLMINI",
  date = "2009-11-10",
  p_attribute = c("word", "pos"),
  interjection = "speech"
)
protocol_cooc <- Cooccurrences(
  protocol,
  p_attribute = c("word", "pos"),
  left = 3L, right = 3L
  )
ll(protocol_cooc)
decode(protocol_cooc)

merkel <- partition(
  "GERMAPARLMINI",
  speaker = "Merkel",
  date = "2009-11-10",
  interjection = "speech",
  regex = TRUE,
  p_attribute = c("word", "pos")
)
merkel_cooc <- Cooccurrences(
  merkel,
  p_attribute = c("word", "pos"),
  left = 3L, right = 3L,
  verbose = TRUE
)
ll(merkel_cooc)
decode(merkel_cooc)

f <- features(merkel_cooc, protocol_cooc)
f <- subset(f, a_pos %in% c("NN", "ADJA"))
f <- subset(f, b_pos %in% c("NN", "ADJA"))
f <- subset(f, c(rep(TRUE, times = 50), rep(FALSE, times = nrow(f) - 50)))

merkel_min <- subset(merkel_cooc, by = f)

if (requireNamespace("igraph", quietly = TRUE)){
  g <- as_igraph(merkel_min, as.undirected = TRUE)
  plot(g)
}
```

```
## End(Not run)
```

Cooccurrences-class *Cooccurrences class for corpus/partition.*

Description

The Cooccurrences-class stores the information for all cooccurrences in a corpus. As this data can be bulky, in-place modifications of the data.table in the stat-slot of a Cooccurrences-object are used wherever possible, to avoid copying potentially large objects whenever possible. The class inherits from the textstat-class, so that methods for textstat-objects are inherited (see examples).

Usage

```
## S4 method for signature 'Cooccurrences'
as.simple_triplet_matrix(x)

## S4 method for signature 'Cooccurrences'
as_igraph(x, edge_attributes = c("ll",
  "ab_count", "rank_ll"), vertex_attributes = "count",
  as.undirected = TRUE, drop = c("\u0084", "\u0093"))

## S4 method for signature 'Cooccurrences'
subset(x, ..., by)

## S4 method for signature 'Cooccurrences'
decode(.Object)

## S4 method for signature 'Cooccurrences'
kwic(.Object,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"), verbose = TRUE,
  progress = TRUE)

## S4 method for signature 'Cooccurrences'
as.sparseMatrix(x, col = "ab_count", ...)
```

Arguments

x	A Cooccurrences class object.
edge_attributes	Attributes from stat data.table in x to add to edges.
vertex_attributes	Vertex attributes to add to nodes.

as.undirected	Logical, whether to return directed or undirected graph.
drop	A character vector indicating names of nodes to drop from igraph object that is prepared.
...	Further arguments passed into a further call of subset.
by	A features-class object.
.Object	A Cooccurrences-class object.
left	Number of tokens to the left of the node.
right	Number of tokens to the right of the node.
verbose	Logical.
progress	Logical, whether to show progress bar.
col	A column to extract.

Details

The as.simple_triplet_matrix-method will transform a Cooccurrences object into a sparse matrix. For reasons of memory efficiency, decoding token ids is performed within the method at the as late as possible. It is NOT necessary that decoded tokens are present in the table in the Cooccurrences object.

The as_igraph-method can be used to turn an object of the Cooccurrences-class into an igraph-object.

The subset method, as a particular feature, allows a Coocccurrences-object to be subsetted by a featurs-Object resulting from a features extraction that compares two Cooccurrences objects.

For reasons of memory efficiency, the initial data.table in the slot stat of a Cooccurrences-object will identify tokens by an integer id, not by the string of the token. The decode()-method will replace these integer columns with human-readable character vectors. Due to the reference logic of the data.table object, this is an in-place operation, peformed without copying the table. The modified object is returned invisibly; usually it will not be necessary to catch the return value.

The kwic-method will add a column to the data.table in the stat-slot with the concordances that are behind a statistical finding, and to the data.table in the stat-slot of the partition in the slot partition. It is an in-place operation.

Returns a sparseMatrix based on the counts of term cooccurrences. At this stage, it is required that decoded tokens are present.

Slots

left Single integer value, number of tokens to the left of the node.

right Single integer value, number of tokens to the right of the node.

p_attribute A character vector, the p-attribute(s) the evaluation of the corpus is based on.

corpus Length-one character vector, the CWB corpus used.

stat A data.table with the statistical analysis of cooccurrences.

encoding Length-one character vector, the encoding of the corpus.

partition The partition that is the basis for computations.

window_sizes A data.table linking the number of tokens in the context of a token identified by id.

minimized Logical, whether the object has been minimized.

See Also

See the documentation of the Cooccurrences-method (including examples) for procedures to get and filter cooccurrence information. See the documentation for the textstat-class explaining which methods for this superclass of the Cooccurrences-class which are available.

Examples

```
X <- Cooccurrences("REUTERS", p_attribute = "word", left = 2L, right = 2L)
m <- as.simple_triplet_matrix(X)
## Not run:
X <- Cooccurrences("REUTERS", p_attribute = "word", left = 5L, right = 5L)
decode(X)
sm <- as.sparseMatrix(X)
stm <- as.simple_triplet_matrix(X)

## End(Not run)
```

---

cooccurrences-class *Cooccurrences class.*

Description

S4 class to organize information of context analysis

Usage

```
## S4 method for signature 'cooccurrences'
show(object)

## S4 method for signature 'cooccurrences_bundle'
as.data.frame(x)

## S4 method for signature 'cooccurrences'
format(x, digits = 2L)

## S4 method for signature 'cooccurrences'
view(.Object)

## S4 method for signature 'cooccurrences_reshaped'
view(.Object)
```

Arguments

object	object to work with
x	object to work with
digits	Integer indicating the number of decimal places (round) or significant digits (signif) to be used.
.Object	object to work with

Slots

`call` Object of class `character` the call that generated the object

`partition` Object of class `character` the partition the analysis is based on

`size_partition` Object of class `integer` the size of the partition

`left` Object of class `integer` number of tokens to the left.

`right` Object of class `integer` number of tokens to the right.

`p_attribute` Object of class `character` p-attribute of the query

`corpus` Object of class `character` the CWB corpus used

`stat` Object of class `data.table` statistics of the analysis

`encoding` Object of class `character` encoding of the corpus

`method` Object of class `character` statistical test(s) used

`Corpus` *Corpus class.*

Description

The R6 `Corpus` class offers a set of methods to retrieve and manage CWB indexed corpora.

Usage

```
Corpus
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`corpus` character vector (length 1), a CWB corpus

`encoding` encoding of the corpus (typically 'UTF-8' or 'latin1'), assigned automatically upon initialization of the corpus

`cpos` a two-column `matrix` with regions of a corpus underlying the s-attributes of the `data.table` in field `s_attributes`

`s_attributes` a `data.table` with the values of a set of s-attributes

`stat` a `data.table` with counts

Arguments

corpus a corpus

registryDir the directory where the registry file resides

dataDir the data directory of the corpus

p_attribute p-attribute, to perform count

s_attributes s-attributes

decode logical, whether to turn token ids into strings upon counting

as.html logical

Methods

`initialize(corpus, p_attribute = NULL, s_attributes = NULL)` Initialize a new object of class `Corpus`.

`count(p_attribute = getOption("polmineR.p_attribute"), decode = TRUE)` Perform counts.

`as.partition()` turn `Corpus` into a partition

`getInfo(as.html = FALSE)`

`showInfo()`

Examples

```
use("polmineR")
REUTERS <- Corpus$new("REUTERS")
infofile <- REUTERS$getInfo()
if (interactive()) REUTERS$showInfo()

# use Corpus class to manage counts
REUTERS <- Corpus$new("REUTERS", p_attribute = "word")
REUTERS$stat

# use Corpus class for creating partitions
REUTERS <- Corpus$new("REUTERS", s_attributes = c("id", "places"))
usa <- partition(REUTERS, places = "usa")
sa <- partition(REUTERS, places = "saudi-arabia", regex = TRUE)

reut <- REUTERS$as.partition()
```

---

corpus *Get corpus/corpora available or used.*

Description

Calling `corpus()` will return a `data.frame` listing the corpora described in the active registry directory, and some basic information on the corpora. If `object` is an object inheriting from the `textstat`, or the `bundle` class, the corpus used to generate the object is returned.

Usage

```
corpus(object)

## S4 method for signature 'textstat'
corpus(object)

## S4 method for signature 'kwic'
corpus(object)

## S4 method for signature 'character'
corpus(object)

## S4 method for signature 'bundle'
corpus(object)

## S4 method for signature 'missing'
corpus()
```

Arguments

object	An object inheriting from the textstat or bundle superclasses.

Examples

```
use("polmineR")
corpus()

p <- partition("REUTERS", places = "kuwait")
corpus(p)

pb <- partition_bundle("REUTERS", s_attribute = "id")
corpus(pb)
```

---

corpus-class *S4 class to wrap information on CWB corpora.*

Description

S4 class to wrap information on CWB corpora.

Usage

```
## S4 method for signature 'corpus'
x$name

## S4 method for signature 'corpus,ANY'
e1 == e2
```

```
## S4 method for signature 'corpus,ANY'
e1 != e2

## S4 method for signature 'corpus'
x %in% table

## S4 method for signature 'corpus'
zoom(x, ...)
```

Arguments

x	A corpus object.
name	An s-attribute that will be assigned as key to a partition.
e1	A first expression, a partition object in this case.
e2	A second expression, the value of an s-attribute.
table	Values a s-attribute shall assume.
...	Further arguments.

Details

The '$'-method will assign the argument name to the slot key and return the modified object.

Slots

corpus A length-one character vector, a CWB corpus.

data_dir The directory where the files for the indexed corpus are.

type The type of the corpus (e.g. "plpr" for a corpus of plenary protocols).

encoding The encoding of the corpus, given as a length-one character vector.

key A length-one character vector stating a s-attribute that serves as a key.

See Also

Other classes to manage corpora: regions, subcorpus-class

Examples

```
corp <- corpus("GERMAPARLMINI")
corp2 <- corp$speaker
corp2@key
x <- corpus("GERMAPARLMINI")
x$date == "2009-10-28"
x <- corpus("GERMAPARLMINI")
x$party != "NA"
x <- corpus("GERMAPARLMINI")
x$date %in% c("2009-10-27", "2009-10-28")
x <- corpus("GERMAPARLMINI")
y <- zoom(x, date == "2009-10-28")
```

```
x <- partition("GERMAPARLMINI", interjection = "speech")
m <- zoom(x, date == "2009-10-28" & speaker == "Angela Dorothea Merkel")

not_unknown <- zoom(x, party != c("NA", "FDP"))
s_attributes(not_unknown, "party")
```

---

count *Get counts.*

Description

Count all tokens, or number of occurrences of a query (CQP syntax may be used), or matches for the query.

Usage

```
count(.Object, ...)

## S4 method for signature 'partition'
count(.Object, query = NULL, cqp = is.cqp,
  check = TRUE, breakdown = FALSE, decode = TRUE,
  p_attribute = getOption("polmineR.p_attribute"),
  mc = getOption("polmineR.cores"), verbose = TRUE, progress = FALSE,
  ...)

## S4 method for signature 'partition_bundle'
count(.Object, query = NULL, cqp = FALSE,
  p_attribute = NULL, freq = FALSE, total = TRUE, mc = FALSE,
  progress = TRUE, verbose = FALSE, ...)

## S4 method for signature 'character'
count(.Object, query = NULL, cqp = is.cqp,
  check = TRUE, p_attribute = getOption("polmineR.p_attribute"),
  breakdown = FALSE, sort = FALSE, decode = TRUE, verbose = TRUE,
  ...)

## S4 method for signature 'vector'
count(.Object, corpus, p_attribute, ...)

## S4 method for signature 'Corpus'
count(.Object, query = NULL, p_attribute)
```

Arguments

.Object	A partition or partition_bundle, or a length-one character vector providing the name of a corpus.

...	Further arguments.
query	A character vector (one or multiple terms), CQP syntax can be used.
cqp	Either logical (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not (defaults to is.query auxiliary function).
check	A logical value, whether to check validity of CQP query using check_cqp_query.
breakdown	Logical, whether to report number of occurrences for different matches for a query.
decode	Logical, whether to turn token ids into decoded strings (only if query is NULL).
p_attribute	The p-attribute(s) to use.
mc	Logical, whether to use multicore (defaults to FALSE).
verbose	Logical, whether to be verbose.
progress	Logical, whether to show progress bar.
freq	Logical, if FALSE, counts will be reported, if TRUE, (relative) frequencies are added to table.
total	Defaults to FALSE, if TRUE, the total value of counts (column named 'TOTAL') will be amended to the data.table that is returned.
sort	Logical, whether to sort table with counts (in stat slot).
corpus	The name of a CWB corpus.

Details

If .Object is a partiton_bundle, the data.table returned will have the queries in the columns, and as many rows as there are in the partition_bundle.

If .Object is a length-one character vector and query is NULL, the count is performed for the whole partition.

If breakdown is TRUE and one query is supplied, the function returns a frequency breakdown of the results of the query. If several queries are supplied, frequencies for the individual queries are retrieved.

Value

A data.table if argument query is used, a count-object, if query is NULL and .Object is a character vector (referring to a corpus) or a partition, a count_bundle-object, if .Object is a partition_bundle.

References

Baker, Paul (2006): *Using Corpora in Discourse Analysis*. London: continuum, p. 47-69 (ch. 3).

See Also

For a metadata-based breakdown of counts (i.e. tabulation by s-attributes), see dispersion.

count

Examples

```
use("polmineR")
debates <- partition("GERMAPARLMINI", date = ".*", regex=TRUE)
count(debates, query = "Arbeit") # get frequencies for one token
count(debates, c("Arbeit", "Freizeit", "Zukunft")) # get frequencies for multiple tokens

count("GERMAPARLMINI", query = c("Migration", "Integration"), p_attribute = "word")

debates <- partition_bundle(
  "GERMAPARLMINI", s_attribute = "date", values = NULL,
  regex = TRUE, mc = FALSE, verbose = FALSE
)
y <- count(debates, query = "Arbeit", p_attribute = "word")
y <- count(debates, query = c("Arbeit", "Migration", "Zukunft"), p_attribute = "word")

count("GERMAPARLMINI", '"Integration.*"', breakdown = TRUE)

P <- partition("GERMAPARLMINI", date = "2009-11-11")
count(P, '"Integration.*"', breakdown = TRUE)
```

`count_class` *Count class.*

Description

S4 class to organize counts. The classes `polmineR` and `ngrams` inherit from the class.

Usage

```
## S4 method for signature 'count'
length(x)

## S4 method for signature 'count'
hist(x, ...)
```

Arguments

`x`	A `count` object, or a class inheriting from `count`.
`...`	Further parameters.

Details

The `length`-method is synonymous with the `size`-method and will return the size of the `corpus` or `partition` a count has been derived from.

Slots

`stat` Object of class `data.table`

`corpus` Object of class `character` the CWB corpus the partition is based on

`encoding` Object of class `character` encoding of the corpus

`name` Object of class `character`, a name for the object

`size` Object of class `integer`, the size of the partition or corpus the count is based upon

Author(s)

Andreas Blaette

See Also

The `count`-class inherits from the `textstat-class`

---

cpos *Get corpus positions for a query or queries.*

Description

Get matches for a query in a CQP corpus, optionally using the CQP syntax of the Corpus Workbench (CWB).

Usage

```
cpos(.Object, ...)

## S4 method for signature 'character'
cpos(.Object, query,
  p_attribute = getOption("polmineR.p_attribute"), cqp = is.cqp,
  check = TRUE, encoding = NULL, verbose = TRUE, ...)

## S4 method for signature 'partition'
cpos(.Object, query, cqp = is.cqp, check = TRUE,
  p_attribute = NULL, verbose = TRUE, ...)

## S4 method for signature 'tempcorpus'
cpos(.Object, query, shift = TRUE)

## S4 method for signature 'matrix'
cpos(.Object)

## S4 method for signature 'hits'
cpos(.Object)
```

Arguments

.Object	A character vector indicating a CWB corpus, a partition object, a tempcorpus object, or a matrix with corpus positions.
...	Used for reasons of backwards compatibility to process arguments that have been renamed (e.g. pAttribute).
query	A character vector providing one or multiple queries (token or CQP query)
p_attribute	The p-attribute to search. Needs to be stated only if query is not a CQP query. Defaults to NULL.
cqp	Either logical (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not (defaults to is.cqp auxiliary function).
check	A logical value, whether to check validity of CQP query using check_cqp_query.
encoding	The encoding of the corpus (if NULL, the encoding stated in the registry file of the corpus will be used),
verbose	A logical value, whether to show messages.
shift	logical, if true, the cpos resulting from the query performed on the tempcorpus will be shifted so that they match the positions of the corpus from which the tempcorpus was generated

Details

If the cpos-method is applied on "character", "partition", or "tempcorpus" object, the result is a two-column matrix with the regions (start end end corpus positions of the matches) for a query. CQP syntax can be used. The encoding of the query is adjusted to conform to the encoding of the CWB corpus.

If the cpos-method is called on a matrix object, the cpos matrix is unfolded, the return value is an integer vector with the individual corpus positions. Equally, if .Object is a hits object, an integer vector is returned with the individual corpus positions.

Value

Unless .Object is a matrix, the return value is a matrix with two columns. The first column reports the left/starting corpus positions (cpos) of the hits obtained. The second column reports the right/ending corpus positions of the respective hit. The number of rows is the number of hits. If there are no hits, a NULL object is returned.

CQI.super *Interfaces for accessing the CWB*

Description

The package offers two different interfaces to the Corpus Workbench (CWB): The package 'RcppCWB', or via cqpserver. An object called 'CQI' will be instantiated in the environment of the polmineR package; the class will provide the functionality to access CWB corpora.

Usage

```
CQI.super

CQI.RcppCWB

CQI.cqpserver

CQI.cqpserver
```

Format

An object of class `R6ClassGenerator` of length 24.

---

cqp *Tools for CQP queries.*

Description

Test whether a character string is a CQP query, or turn a character vector into CQP queries.

Usage

```
is.cqp(query)

check_cqp_query(query)

as.cqp(query, normalise.case = FALSE, collapse = FALSE)
```

Arguments

`query`	A `character` vector with at least one CQP query.
`normalise.case`	A `logical` value, if `TRUE`, a flag will be added to the query/queries to omit matching case.
`collapse`	A `logical` value, whether to collapse the queries into one.

Details

The `is.cqp` function guesses whether `query` is a CQP query and returns the respective logical value (`TRUE`/`FALSE`).

The `as.cqp` function takes a character vector as input and converts it to a CQP query by putting the individual strings in quotation marks.

The `check_cqp_query`-function will check that opening quotation marks are matched by closing quotation marks, to prevent crashes of CQP and the R session.

Value

is.cqp returns a logical value, as.cqp a character vector, check_cqp_query a logical value that is TRUE if all queries are valid, or FALSE if not.

References

CQP Query Language Tutorial (http://cwb.sourceforge.net/files/CQP_Tutorial.pdf)

Examples

```
is.cqp("migration") # will return FALSE
is.cqp('"migration"') # will return TRUE
is.cqp('[pos = "ADJA"] "migration"') # will return TRUE

as.cqp("migration")
as.cqp(c("migration", "diversity"))
as.cqp(c("migration", "diversity"), collapse = TRUE)
as.cqp("migration", normalise.case = TRUE)


check_cqp_query('"Integration.*"') # TRUE, the query is ok
check_cqp_query('"Integration.*') # FALSE, closing quotation mark is missing
check_cqp_query("'Integration.*") # FALSE, closing quotation mark is missing
check_cqp_query(c("'Integration.*", '"Integration.*')) # FALSE too
```

---

decode *Decode structural attribute, partition or corpus.*

Description

Function that can be applied on a corpus or a partition. The returned data.table can be coerced to a tibble easily and processed according to tidytext approaches.

Usage

```
decode(.Object, s_attribute = NULL, verbose = TRUE, ...)

decode(.Object, s_attribute = NULL, verbose = TRUE, ...)
```

Arguments

.Object	The corpus or partition to decode (character vector).
s_attribute	The s-attribute to decode.
verbose	Logical value, whether to output messages.
...	Further arguments.

Details

If a s_attribute is a character vector providing one or several structural attributes, the return value is a data.table with the left and right corpus positions in the first and second columns ("cpos_left" and "cpos_right"). Values of further columns are the decoded s-attributes. The name of the s-attribute is the column name. An error is thrown if the lengths of structural attributes differ (i.e. if there is a nested data structure).

If s_attribute is NULL, the token stream is decoded for all positional attributes that are present. Structural attributes are reported in additional columns. Decoding the entire corpus may be useful to make a transition to processing data following the 'tidy' approach, or to manipulate the corpus data and to re-encode the corpus.

Value

The return value is a data.table.

Examples

```
## Not run:
use("polmineR")

# Scenario 1: Decode one or two s-attributes
dt <- decode("REUTERS", s_attribute = "id")
dt <- decode("REUTERS", s_attribute = c("topics_cat", "places"))

# Scenario 2: Decode entire corpus
dt <- decode("REUTERS")

# Scenario 3: Decode partition
p <- partition("REUTERS", places = "kuwait", regex = TRUE)
dt <- decode(p)

# Scenario 4: Decode partition_bundle
pb <- partition_bundle("REUTERS", s_attribute = "id")
dts <- lapply(as.list(pb), decode)
dts <- lapply(names(dts), function(n) dts[[n]][, speech_id := n])
dt <- data.table::rbindlist(dts)

## End(Not run)
```

---

dispersion *Dispersion of a query or multiple queries*

Description

The function returns the frequencies of a query or a multiple queries in sub-partitions defined by one or two dimensions. This is a wrapper function, so the output will depend on the number of queries and dimensions provided.

Usage

```
dispersion(.Object, ...)

## S4 method for signature 'partition'
dispersion(.Object, query, s_attribute,
  cqp = FALSE, p_attribute = getOption("polmineR.p_attribute"),
  freq = FALSE, mc = FALSE, progress = TRUE, verbose = FALSE, ...)

## S4 method for signature 'character'
dispersion(.Object, query, s_attribute,
  cqp = is.cqp, p_attribute = getOption("polmineR.p_attribute"),
  freq = FALSE, mc = FALSE, progress = TRUE, verbose = TRUE, ...)

## S4 method for signature 'hits'
dispersion(.Object, s_attribute, freq = FALSE,
  verbose = TRUE, ...)
```

Arguments

.Object	a partition object
...	further parameters
query	a character vector containing one or multiple queries
s_attribute	a character vector of length 1 or 2 providing the s-attributes
cqp	if logical, whether the query is a CQP query (TRUE/FALSE), if it is a function that is passed in, the function will be applied to the query to guess whether query is a CQP query
p_attribute	the p-attribute that will be looked up, typically 'word' or 'lemma'
freq	logical, whether to calculate normalized frequencies
mc	logical, whether to use multicore
progress	logical, whether to shop progress
verbose	logical, whether to be verbose

Value

depends on the input, as this is a wrapper function

Author(s)

Andreas Blaette

See Also

crosstab-class

count

Examples

```
use("polmineR")
test <- partition("GERMAPARLMINI", date = ".*", p_attribute = NULL, regex = TRUE)
integration <- dispersion(
  test, query = "Integration",
  p_attribute = "word", s_attribute = "date"
)
integration <- dispersion(test, "Integration", s_attribute = c("date", "party"))
integration <- dispersion(test, '"Integration.*"', s_attribute = "date", cqp = TRUE)
```

`dotplot` *dotplot*

Description

dotplot

Usage

```
dotplot(.Object, ...)

## S4 method for signature 'textstat'
dotplot(.Object, col, n = 20L, ...)

## S4 method for signature 'features'
dotplot(.Object, col = NULL, n = 20L, ...)

## S4 method for signature 'features_ngrams'
dotplot(.Object, col = NULL, n = 20L, ...)

## S4 method for signature 'partition'
dotplot(.Object, col = "count", n = 20L, ...)
```

Arguments

`.Object`	object
`...`	further arguments that will be passed into the `dotchart` function
`col`	column
`n`	number

encoding *Get and set encoding.*

Description

Method for textstat objects and classes inheriting from textstat; if object is a character vector, the encoding of the corpus is returned..

Usage

```
encoding(object)

encoding(object) <- value

## S4 method for signature 'textstat'
encoding(object)

## S4 method for signature 'bundle'
encoding(object)

## S4 method for signature 'character'
encoding(object)
```

Arguments

object	A textstat or bundle object, or a length-one character vector specifying a corpus.
value	Value to be assigned.

Examples

```
# Get encoding of a corpus.
encoding("REUTERS")

# Get encoding of a partition.
r <- partition("REUTERS", places = "kuwait", regex = TRUE)
encoding(r)

# Get encoding of another class inheriting from textstat (count).
cnt <- count("REUTERS", p_attribute = "word")
encoding(cnt)

# Get encoding of objects in a bundle.
pb <- partition_bundle("REUTERS", s_attribute = "id")
encoding(pb)
```

encodings *Conversion between corpus and native encoding.*

Description

Utility functions to convert encoding between the native encoding and the encoding of the corpus.

Usage

```
as.utf8(x, from)

as.nativeEnc(x, from)

as.corpusEnc(x, from = localeToCharset()[1], corpusEnc)
```

Arguments

x	the object (a character vector)
from	encoding of the input character vector
corpusEnc	encoding of the corpus (e.g. "latin1", "UTF-8")

Details

The encoding of a corpus and the encoding of the terminal (the native encoding) may differ and evoke strange output, or wrong results if no conversion is carried out between the potentially differing encodings. The functions `as.nativeEnc` and `as.corpusEnc` are auxiliary functions to assist this. The functions `as.nativeEnc` and `as.utf8` deliberately remove the explicit statement of the encoding, to avoid warnings that may occur with character vector columns in a `data.table` object.

enrich *Enrich an object.*

Description

Methods to enrich objects with additional (statistical) information. The methods are documented with the classes to which they adhere. See the references in the `seealso`-section.

Usage

```
enrich(.Object, ...)
```

Arguments

.Object	a `partition`, `partition_bundle` or comp object
...	further parameters

See Also

The enrich method is defined for the following classes: "partition", (see partition_class), "partition_bundle" (see partition_bundle-class), "kwic" (see kwic-class), and "context" (see context-class). See the linked documentation to learn how the enrich method can be applied to respective objects.

`features` *Get features by comparison.*

Description

The features of two objects, usually a `partition` defining a corpus of interest (coi), and a `partition` defining a reference corpus (ref) are compared. The most important purpose is term extraction.

Usage

```
features(x, y, ...)

## S4 method for signature 'partition'
features(x, y, included = FALSE,
  method = "chisquare", verbose = FALSE)

## S4 method for signature 'count'
features(x, y, by = NULL, included = FALSE,
  method = "chisquare", verbose = TRUE)

## S4 method for signature 'partition_bundle'
features(x, y, included = FALSE,
  method = "chisquare", verbose = TRUE,
  mc = getOption("polmineR.mc"), progress = FALSE)

## S4 method for signature 'ngrams'
features(x, y, included = FALSE,
  method = "chisquare", verbose = TRUE, ...)

## S4 method for signature 'Cooccurrences'
features(x, y, included = FALSE,
  method = "ll", verbose = TRUE)
```

Arguments

`x`	A `partition` or `partition_bundle` object.
`y`	A `partition` object, it is assumed that the coi is a subcorpus of ref
`...`	further parameters
`included`	TRUE if coi is part of ref, defaults to FALSE
`method`	the statistical test to apply (chisquare or log likelihood)

verbose	A logical value, defaults to TRUE
by	the columns used for merging, if NULL (default), the p-attribute of x will be used
mc	logical, whether to use multicore
progress	logical

Author(s)

Andreas Blaette

References

Baker, Paul (2006): *Using Corpora in Discourse Analysis*. London: continuum, p. 121-149 (ch. 6).

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 151-189 (ch. 5).

Examples

```
use("polmineR")

kauder <- partition(
  "GERMAPARLMINI",
  speaker = "Volker Kauder", interjection = "speech",
  p_attribute = "word"
  )
all <- partition("GERMAPARLMINI", interjection = "speech", p_attribute = "word")

terms_kauder <- features(x = kauder, y = all, included = TRUE)
top100 <- subset(terms_kauder, rank_chisquare <= 100)
head(top100)

# a different way is to compare count objects
kauder_count <- as(kauder, "count")
all_count <- as(all, "count")
terms_kauder <- features(kauder_count, all_count, included = TRUE)
top100 <- subset(terms_kauder, rank_chisquare <= 100)
head(top100)

speakers <- partition_bundle("GERMAPARLMINI", s_attribute = "speaker")
speakers <- enrich(speakers, p_attribute = "word")
speaker_terms <- features(speakers[[1:5]], all, included = TRUE, progress = TRUE)
dtm <- as.DocumentTermMatrix(speaker_terms, col = "chisquare")
```

features-class *Feature selection by comparison.*

Description

The features-method returns a features-object. Several features-objects can be combined into a features_bundle-object.

Usage

```
## S4 method for signature 'features'
summary(object)

## S4 method for signature 'features'
show(object)

## S4 method for signature 'features_bundle'
summary(object)

## S4 method for signature 'features'
format(x, digits = 2L)

## S4 method for signature 'features'
view(.Object)
```

Arguments

object	A features or features_bundle object.
x	A features object.
digits	Integer indicating the number of decimal places (round) or significant digits (signif) to be used.
.Object	a features object.

Details

A set of features objects can be combined into a features_bundle. Typically, a features_bundle will result from applying the features-method on a partition_bundle. See the documentation for bundle to learn about the methods for bundle objects that are available for a features_bundle.

Slots

corpus The CWB corpus the features are derived from, a character vector of length 1.

p_attribute Object of class character.

encoding Object of class character.

corpus Object of class character.

`stat` Object of class `data.frame`.

`size_coi` Object of class `integer`.

`size_ref` Object of class `integer`.

`included` Object of class `logical` whether corpus of interest is included in reference corpus

`method` Object of class `character` statisticalTest used

`call` Object of class `character` the call that generated the object

Author(s)

Andreas Blaette

`get_template` *Get and set templates.*

Description

Templates are used to format the markdown/html output of partitions. Upon loading the polmineR package, templates for corpora are loaded into the option 'polmineR.templates'.

Usage

```
get_template(.Object, ...)

## S4 method for signature 'character'
get_template(.Object)

## S4 method for signature 'partition'
get_template(.Object)

## S4 method for signature 'missing'
get_template(.Object)

set_template(.Object, ...)

## S4 method for signature 'character'
set_template(.Object)

## S4 method for signature 'missing'
set_template(.Object, verbose = FALSE)
```

Arguments

`.Object`	object
`...`	further parameters
`verbose`	logical, whether to be verbose

get_token_stream *Get Token Stream Based on Corpus Positions.*

Description

Turn regions of a corpus defined by corpus positions into the original text.

Usage

```
get_token_stream(.Object, ...)

## S4 method for signature 'numeric'
get_token_stream(.Object, corpus, p_attribute,
  encoding = NULL, collapse = NULL, beautify = TRUE, cpos = FALSE,
  cutoff = NULL, decode = TRUE, ...)

## S4 method for signature 'matrix'
get_token_stream(.Object, ...)

## S4 method for signature 'character'
get_token_stream(.Object, left = NULL,
  right = NULL, ...)

## S4 method for signature 'partition'
get_token_stream(.Object, p_attribute,
  collapse = NULL, cpos = FALSE, ...)

## S4 method for signature 'regions'
get_token_stream(.Object, p_attribute = "word", ...)
```

Arguments

.Object	An object of class matrix or partition
...	Further arguments.
corpus	The CWB corpus.
p_attribute	The p-attribute to decode.
encoding	Encoding to use.
collapse	Length-one character string.
beautify	Logical, whether to adjust whitespace before and after interpunctation.
cpos	Logical, whether to return cpos as names of the tokens.
cutoff	Maximum number of tokens to be reconstructed.
decode	Logical, whether to decode token ids to character strings.
left	Left corpus position.
right	Right corpus position.

Examples

```
get_token_stream(0:9, corpus = "GERMAPARLMINI", p_attribute = "word")
get_token_stream(0:9, corpus = "GERMAPARLMINI", p_attribute = "word", collapse = " ")
fulltext <- get_token_stream("GERMAPARLMINI", p_attribute = "word")
```

---

get_type *Get corpus/partition type.*

Description

To generate fulltext output, different templates can be used with a behavior that depends on the type of a corpus. get_type will return the type of corpus if it is a specialized one, or NULL.

Usage

```
get_type(.Object)

## S4 method for signature 'character'
get_type(.Object)

## S4 method for signature 'Corpus'
get_type(.Object)

## S4 method for signature 'partition'
get_type(.Object)

## S4 method for signature 'partition_bundle'
get_type(.Object)
```

Arguments

.Object	A partition, partition_bundle, Corpus object, or a length-one character vector indicating a CWB corpus.

Details

When generating a partition, the corpus type will be prefixed to the class that is generated (separated by underscore). If the corpus type is not NULL, a class inheriting from the partition-class is instantiated. Note that at this time, only plpr_partition and press_partition is implemented.

Examples

```
use("polmineR")

get_type("GERMAPARLMINI")

p <- partition("GERMAPARLMINI", date = "2009-10-28")
get_type(p)
```

```
is(p)

pb <- partition_bundle("GERMAPARLMINI", s_attribute = "date")
get_type(pb)

gp <- Corpus$new("GERMAPARLMINI")
get_type(gp)

get_type("REUTERS") # returns NULL - no specialized corpus
```

---

highlight *Highlight tokens in text output.*

Description

Highlight tokens in fulltext based on exact match, a regular expression or corpus position in `kwic` output or `html` document.

Usage

```
highlight(.Object, ...)

## S4 method for signature 'character'
highlight(.Object, highlight = list(), ...)

## S4 method for signature 'html'
highlight(.Object, highlight = list(), ...)

## S4 method for signature 'kwic'
highlight(.Object, highlight = list(), regex = FALSE,
  perl = TRUE, verbose = TRUE, ...)
```

Arguments

`.Object`	A `html`, `character`, a `kwic` object.
`...`	Terms to be highlighted can be passed in as named character vectors of terms (or regular expressions); the name then needs to be a valid color name.
`highlight`	A character vector, or a list of character or integer vectors.
`regex`	Logical, whether character vectors are interpreted as regular expressions.
`perl`	Logical, whether to use perl-style regular expressions for highlighting when regex is `TRUE`.
`verbose`	Logical, whether to output messages.

Details

If highlight is a character vector, the names of the vector are interpreted as colors. If highlight is a list, the names of the list are considered as colors. Values can be character values or integer values with token ids. Colors are inserted into the output html and need to be digestable for the browser used.

Examples

```
use("polmineR")
P <- partition("REUTERS", places = "argentina")
H <- html(P)
Y <- highlight(H, list(lightgreen = "higher"))
if (interactive()) htmltools::html_print(Y)

# highlight matches for a CQP query
H2 <- highlight(
  H,
  highlight = list(yellow = cpos(hits(P, query = '"prod.*"', cqp = TRUE)))
)

# the method can be used in pipe
if (require("magrittr")){
  P %>% html() %>% highlight(list(lightgreen = "1986")) -> H
  P %>% html() %>% highlight(list(lightgreen = c("1986", "higher"))) -> H
  P %>% html() %>% highlight(list(lightgreen = 4020:4023)) -> H
}

# use highlight for kwic output
K <- kwic("REUTERS", query = "barrel")
K2 <- highlight(K, highlight = list(yellow = c("oil", "price")))
if (interactive()) K2

# use character vector for output, not list
K2 <- highlight(
  K,
  highlight = c(
    green = "pric.",
    red = "reduction",
    red = "decrease",
    orange = "dropped"),
    regex = TRUE
)
if (interactive()) K2
```

---

hits *Get Hits.*

Description

Get hits for a (set of) queries, optionally with s-attribute values.

Usage

```
hits(.Object, ...)

## S4 method for signature 'character'
hits(.Object, query, cqp = FALSE, check = TRUE,
  s_attribute = NULL, p_attribute = "word", size = FALSE,
  freq = FALSE, mc = FALSE, verbose = TRUE, progress = TRUE, ...)

## S4 method for signature 'partition'
hits(.Object, query, cqp = FALSE,
  s_attribute = NULL, p_attribute = "word", size = FALSE,
  freq = FALSE, mc = FALSE, progress = FALSE, verbose = TRUE, ...)

## S4 method for signature 'partition_bundle'
hits(.Object, query, cqp = FALSE,
  check = TRUE, p_attribute = getOption("polmineR.p_attribute"),
  size = TRUE, freq = FALSE, mc = getOption("polmineR.mc"),
  progress = FALSE, verbose = TRUE, ...)

## S4 method for signature 'context'
hits(.Object, s_attribute = NULL, verbose = TRUE,
  ...)
```

Arguments

.Object	a character, partition or partition_bundle object
...	further parameters
query	a (optionally named, see datails) character vector with one or more queries
cqp	either logical (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not
check	A logical value, whether to check validity of CQP query using check_cqp_query.
s_attribute	s-attributes
p_attribute	p-attribute
size	logical - return size of subcorpus
freq	logcial - return relative frequencies
mc	logical, whether to use multicore
verbose	logical
progress	logical, whether to show progress bar

Details

If the query character vector is named, the names of the query occurr in the data.table that is returned rather than the queries.

If freq is TRUE, the data.table returned in the DT-slot will deliberately include the subsets of the partition/corpus with no hits (query is NA, count is 0).

hits_class *Hits class.*

Description

Hits class.

Usage

```
## S4 method for signature 'hits'
sample(x, size)
```

Arguments

x	A hits object.
size	A non-negative integer giving the number of items to choose.

Slots

stat a "data.table"

corpus a "character" vector

query Object of class "character"

p_attribute p-attribute that has been queried

encoding encoding of the corpus

name name of the object

html *Generate html from object.*

Description

Prepare a html document to inspect the full text.

Usage

```
html(object, ...)

## S4 method for signature 'character'
html(object)

## S4 method for signature 'partition'
html(object, meta = NULL, cpos = TRUE,
  verbose = FALSE, cutoff = NULL, charoffset = FALSE,
  beautify = TRUE, height = NULL, ...)
```

```
## S4 method for signature 'partition_bundle'
html(object, filename = c(),
  type = "debate")

## S4 method for signature 'kwic'
html(object, i, s_attribute = NULL, type = NULL,
  verbose = FALSE, ...)

## S3 method for class 'html'
print(x, ...)
```

Arguments

object	the object the fulltext output will be based on
...	further parameters that are passed into as.markdown
meta	metadata for output, if NULL (default), the s-attributes defining a partition will be used
cpos	logical, if TRUE (default), all tokens will be wrapped by elements with id attribute indicating corpus positions
verbose	logical, whether to be verbose
cutoff	maximum number of tokens to decode from token stream, passed into as.markdown
charoffset	Logical, if TRUE, character offset positions are added to elements embracing tokens.
beautify	Logical, if TRUE, whitespace before interpunctuation will be removed.
height	A character vector that will be inserted into the html as an optional height of a scroll box.
filename	the filename
type	the partition type
i	if object is a kwic-object, the index of the concordance for which the fulltext is to be generated
s_attribute	structural attributes that will be used to define the partition where the match occurred
x	object of class html to print

Details

If param charoffset is TRUE, character offset positions will be added to tags that embrace tokens. This may be useful, if exported html document is annotated with a tools that stores annotations with character offset positions.

Examples

```
use("polmineR")
P <- partition("REUTERS", places = "argentina")
H <- html(P)
if (interactive()) H # show full text in viewer pane

# html-method can be used in a pipe
if (require("magrittr")){
  H <- partition("REUTERS", places = "argentina") %>% html()
  # use html-method to get from concordance to full text
  K <- kwic("REUTERS", query = "barrels")
  H <- html(K, i = 1, s_attribute = "id")
  H <- html(K, i = 2, s_attribute = "id")
  for (i in 1:length(K)) {
    H <- html(K, i = i, s_attribute = "id")
    if (interactive()){
      show(H)
      userinput <- readline("press 'q' to quit or any other key to continue")
      if (userinput == "q") break
    }
  }
}
```

---

kwic *KWIC/concordance output.*

Description

Prepare and show concordances / keyword-in-context (kwic).

Usage

```
kwic(.Object, ...)

## S4 method for signature 'context'
kwic(.Object,
  s_attributes = getOption("polmineR.meta"), cpos = TRUE,
  verbose = FALSE, ...)

## S4 method for signature 'partition'
kwic(.Object, query, cqp = is.cqp,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"),
  s_attributes = getOption("polmineR.meta"), p_attribute = "word",
  boundary = NULL, cpos = TRUE, stoplist = NULL,
  positivelist = NULL, regex = FALSE, verbose = TRUE, ...)
```

```
## S4 method for signature 'character'
kwic(.Object, query, cqp = is.cqp, check = TRUE,
  left = as.integer(getOption("polmineR.left")),
  right = as.integer(getOption("polmineR.right")),
  s_attributes = getOption("polmineR.meta"), p_attribute = "word",
  boundary = NULL, cpos = TRUE, stoplist = NULL,
  positivelist = NULL, regex = FALSE, verbose = TRUE,
  progress = TRUE, ...)
```

Arguments

.Object	A (length-one) character vector with the name of a CWB corpus, a partition or context object.
...	Further arguments, used to ensure backwards compatibility.
s_attributes	Structural attributes (s-attributes) to include into output table as metainformation.
cpos	Logical, if TRUE, the corpus positions ("cpos") if the hits will be included in the kwic-object that is returned.
verbose	Logical, whether to output progress messages
query	A query, CQP-syntax can be used.
cqp	Either a logical value (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not (defaults to auxiliary function is.query).
left	Number of tokens to the left of query match.
right	Number of tokens to the right of query match.
p_attribute	The p-attribute, defaults to 'word'.
boundary	If provided, a length-one character vector stating an s-attribute that will be used to check the boundaries of the text.
stoplist	Terms or ids to prevent a concordance from occurring in results.
positivelist	Terms or ids required for a concordance to occurr in results
regex	Logical, whether stoplist/positivelist is interpreted as regular expression
check	A logical value, whether to check validity of CQP query using check_cqp_query.
progress	Logical, whether to show progress bars.

Details

The method works with a whole CWB corpus defined by a character vector, and can be applied on a partition- or a context object.

If a positivelist ist supplied, only concordances will be kept if at least one of the terms from the positivelist occurrs in the context of the query match. Use argument regex if the positivelist should be interpreted as regular expressions. Tokens from the positivelist will be highlighted in the output table.

References

Baker, Paul (2006): *Using Corpora in Discourse Analysis*. London: continuum, pp. 71-93 (ch. 4).

Jockers, Matthew L. (2014): *Text Analysis with R for Students of Literature*. Cham et al: Springer, pp. 73-87 (chs. 8 & 9).

See Also

The return value is a `kwic-class` object; the documentation for the class explains the methods applicable to `kwic-class` objects. To read the whole text, see the `read`-method.

Examples

```
use("polmineR")
K <- kwic("GERMAPARLMINI", "Integration")
K <- kwic(
  "GERMAPARLMINI",
  "Integration", left = 20, right = 20,
  s_attributes = c("date", "speaker", "party")
)
K <- kwic(
  "GERMAPARLMINI",
  '"Integration" [] "(Menschen|Migrant.*|Personen)"', cqp = TRUE,
  left = 20, right = 20,
  s_attributes = c("date", "speaker", "party")
)

K <- kwic(
  "GERMAPARLMINI",
  '"Sehr" "geehrte"', cqp = TRUE,
  boundary = "date"
)

P <- partition("GERMAPARLMINI", date = "2009-11-10")
K <- kwic(P, query = "Integration")
K <- kwic(P, query = '"Sehr" "geehrte"', cqp = TRUE, boundary = "date")
```

`kwic-class` *kwic (S4 class)*

Description

S4 class for organizing information for kwic/concordance output. A set of standard generics (`show`, `as.character`, `as.data.frame`, `length`, `sample`, `subset`) as well as indexing is implemented to process kwic class objects (see 'Usage'). See section 'Details' for the `enrich`, `view` and `knit_print` methods.

Usage

```
## S4 method for signature 'kwic'
show(object)

## S4 method for signature 'kwic'
knit_print(x,
  pagelength = getOption("polmineR.pagelength"),
  options = knitr::opts_chunk, ...)

## S4 method for signature 'kwic'
as.character(x, fmt = "<i>%s</i>")

## S4 method for signature 'kwic,ANY,ANY,ANY'
x[i]

## S4 method for signature 'kwic'
subset(x, ...)

## S4 method for signature 'kwic'
as.data.frame(x)

## S4 method for signature 'kwic'
length(x)

## S4 method for signature 'kwic'
sample(x, size)

## S4 method for signature 'kwic'
enrich(.Object, s_attributes = NULL, table = FALSE,
  ...)

## S4 method for signature 'kwic'
view(.Object)
```

Arguments

object	A kwic class object.
x	A kwic class object.
pagelength	The number of kwic lines displayed per page in the datatables htmlwidget that is returned.
options	Chunk options.
...	Used for backwards compatibility.
fmt	A format string passed into sprintf to format the node of a KWIC display.
i	Single integer value, the kwic line for which the fulltext shall be inspected.
size	An integer, subset size for sampling.
.Object	A kwic class object.

s_attributes	Character vector of s-attributes with metainformation.
table	Logical, whether to turn cpos data.table into data.frame for output.

Details

The knit_print will be called by knitr when processing code chunks in Rmarkdown documents to include a htmlwidget into the resulting html document. It may be necessary to explicitly state "render=knit_print" in the chunk options.

The subset-method will apply subset to the table in the slot table, for filtering query results based on metadata (i.e. s-attributes) that need to be present.

The enrich method is used to generate the actual output for the kwic method. If param table is TRUE, corpus positions will be turned into a data.frame with the concordance lines. If param s_attributes is a character vector with s-attributes, the respective s-attributes will be added as columns to the table with concordance lines.

Slots

metadata A character vector with s-attributes of the metadata that are to be displayed.

p_attribute The p-attribute for which the context has been generated.

left An integer value, words to the left of the query match.

right An integer value, words to the right of the query match.

corpus Length-one character vector, the CWB corpus.

cpos A data.table with the columns "hit_no", "cpos", "position", "word_id", "word" and "direction".

table A data.frame, a table with columns "left", "node", "right", and metadata, if the object has been enriched.

encoding A length-one character vector with the encoding of the corpus.

labels A character vector with labels.

categories A character vector.

See Also

The constructor for generating kwic objects is the kwic method.

Examples

```
use("polmineR")
K <- kwic("GERMAPARLMINI", "Integration")
length(K)
K_min <- K[1]
K_min <- K[1:5]
oil <- kwic("REUTERS", query = "oil")
as.character(oil)
```

label *Assign and get labels.*

Description

Assign and get labels.

Usage

```
label(x, ...)

label(x) <- value

## S4 method for signature 'kwic'
label(x, n = NULL)
```

Arguments

x	object
...	further parameters
value	length (character vector, length 1)
n	label index

Labels-class *Labels class.*

Description

Labels class.

Arguments

n	length of character vector in field labels
choices	choices to be assigned to field choices
expandable	whether choices are expandable

Fields

labels character vector with labels; if logical or numeric labels are intended, assign them as character vector anyway

choices character vector, a list of choices for labels

expandable whether choices may be expanded (logical)

ll *Compute Log-likelihood Statistics.*

Description

Apply the log-likelihood statistic to detect cooccurrences or keywords.

Usage

```
ll(.Object, ...)

## S4 method for signature 'features'
ll(.Object)

## S4 method for signature 'context'
ll(.Object)

## S4 method for signature 'cooccurrences'
ll(.Object)

## S4 method for signature 'Cooccurrences'
ll(.Object, verbose = TRUE)
```

Arguments

.Object	An object of class cooccurrence, context, or features.
...	Further arguments (such as verbose).
verbose	Logical, whether to output messages.

Details

The log-likelihood test to detect cooccurrences is a standard approach to find collocations (Dunning 1993, Evert 2005, 2009).

(a) The basis for computing for the log-likelihood statistic is a contingency table of observationes, which is prepared for every single token in the corpus. It reports counts for a token to inspect and all other tokens in a corpus of interest (coi) and a reference corpus (ref):

	coi	ref	TOTAL
count token	o_{11}	o_{12}	r_1
other tokens	o_{21}	o_{22}	r_2
TOTAL	c_1	c_2	N

(b) Based on the contingency table(s) with observed counts, expected values are calculated for each cell, as the product of the column and margin sums, divided by the overall number of tokens (see example).

(c) The standard formula for calculating the log-likelihood test is as follows.

$$G^2 = 2 \sum O_{ij} log(\frac{O_{ij}}{E_{ij}})$$

Note: Before polmineR v0.7.11, a simplification of the formula was used (Rayson/Garside 2000), which omits the third and fourth term of the previous formula:

$$ll = 2(o_{11} log(\frac{o_{11}}{E_{11}}) + o_{12} log(\frac{o_{12}}{E_{12}}))$$

There is a (small) gain of computational efficiency using this simplified formula and the result is almost identical with the standard formula; see however the critical discussion of Ulrike Tabbert (2015: 84ff).

The implementation in the `ll`-method uses a vectorized approach of the computation, which is substantially faster than iterating the rows of a table, generating individual contingency tables etc. As using the standard formula is not significantly slower than relying on the simplified formula, polmineR has moved to the standard computation.

An inherent difficulty of the log likelihood statistic is that it is not possible to compute the statistical test value if the number of observed counts in the reference corpus is 0, i.e. if a term only occurrs exclusively in the neighborhood of a node word. When filtering out rare words from the result table, respective `NA` values will usually disappear.

References

Dunning, Ted (1993): Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, Vol. 19, No. 1, pp. 61-74.

Rayson, Paul; Garside, Roger (2000): Comparing Corpora using Frequency Profiling. *The Workshop on Comparing Corpora*. http://aclweb.org/anthology/W00-0901.

Evert, Stefan (2005): *The Statistics of Word Cooccurrences. Word Pairs and Collocations.* URN urn:nbn:de:bsz:93-opus-23714. https://elib.uni-stuttgart.de/bitstream/11682/2573/1/Evert2005phd.pdf

Evert, Stefan (2009). Corpora and Collocations. In: A. Ludeling and M. Kyto (eds.), *Corpus Linguistics. An International Handbook*. Mouton de Gruyter, Berlin, pp. 1212-1248 (ch. 58).

Tabbert, Ulrike (2015): *Crime and Corpus. The Linguistic Representation of Crime in the Press*. Amsterdam: Benjamins.

Examples

```
# use ll-method explicitly
oil <- cooccurrences("REUTERS", query = "oil", method = NULL)
oil <- ll(oil)
oil_min <- subset(oil, count_coi >= 3)
if (interactive()) View(format(oil_min))
summary(oil)

# use ll-method on 'Cooccurrences'-object
R <- Cooccurrences("REUTERS", left = 5L, right = 5L, p_attribute = "word")
ll(R)
```

```
decode(R)
summary(R)

# use log likelihood test for feature extraction
x <- partition(
  "GERMAPARLMINI", speaker = "Merkel",
  interjection = "speech", regex = TRUE,
  p_attribute = "word"
)
f <- features(x, y = "GERMAPARLMINI", included = TRUE, method = "ll")
f <- features(x, y = "GERMAPARLMINI", included = TRUE, method = NULL)
f <- ll(f)
summary(f)

## Not run:

# A sample do-it-yourself calculation for log-likelihood:
# Compute ll-value for query "oil", and "prices"

oil <- context("REUTERS", query = "oil", left = 5, right = 5)

# (a) prepare matrix with observed values
o <- matrix(data = rep(NA, 4), ncol = 2)
o[1,1] <- as(oil, "data.table")[word == "prices"][["count_coi"]]
o[1,2] <- count("REUTERS", query = "prices")[["count"]] - o[1,1]
o[2,1] <- size(oil)[["coi"]] - o[1,1]
o[2,2] <- size(oil)[["ref"]] - o[1,2]


# (b) prepare matrix with expected values, calculate margin sums first
r <- rowSums(o)
c <- colSums(o)
N <- sum(o)

e <- matrix(data = rep(NA, 4), ncol = 2) # matrix with expected values
e[1,1] <- r[1] * (c[1] / N)
e[1,2] <- r[1] * (c[2] / N)
e[2,1] <- r[2] * (c[1] / N)
e[2,2] <- r[2] * (c[2] / N)


# (c) compute log-likelihood value
ll_value <- 2 * (
  o[1,1] * log(o[1,1] / e[1,1]) +
  o[1,2] * log(o[1,2] / e[1,2]) +
  o[2,1] * log(o[2,1] / e[2,1]) +
  o[2,2] * log(o[2,2] / e[2,2])
)

df <- as.data.frame(cooccurrences("REUTERS", query = "oil"))
subset(df, word == "prices")[["ll"]]

## End(Not run)
```

mail *Send the result of an analysis by Email.*

Description

Send out a mail with the statistical analysis included in an object attached as an xlsx-file.

Usage

```
mail(.Object, ...)

## S4 method for signature 'textstat'
mail(.Object, to = getOption("polmineR.email"),
  rows = 1L:min(250L, nrow(.Object)))

## S4 method for signature 'data.frame'
mail(.Object, to = getOption("polmineR.email"),
  filename = tempfile(fileext = ".xlsx"), rows = 1L:min(250L,
  nrow(.Object)))

## S4 method for signature 'kwic'
mail(.Object, to = getOption("polmineR.email"),
  rows = 1L:min(250L, nrow(.Object)))
```

Arguments

.Object	The object to deliver.
...	Further parameters.
to	An email-address, the recipient of the mail message.
rows	The number of rows of the table included in the Excel file to be sent (if NULL, the whole table will be sent).
filename	The filename of the (temporary) xlsx-file that is generated.

Details

The method translates the result table in the object provided into an Excel sheet and attaches the sheet to an Email which will be sent to the Email-address provided by the argument to. A pre-requirement is that the global options polmineR.smtp_port and polmineR.smtp_server are validly defined. See examples.

Please note: At this stage, authentication is not yet supported.

Examples

```
# Get all (global) options for the polmineR package
grep("polmineR", names(options()), value = TRUE)
```

```
# Get options that need to be set
getOption("polmineR.email")
getOption("polmineR.smtp_server")
getOption("polmineR.smtp_port")

# Sample options (let us imagine Donald Duck had a mail-account)
options("polmineR.email" = "donald.duck@duckmail.org")
options("polmineR.smtp_port" = "587")
options("polmineR.smtp_server" = "smtp.duckmail.org")

# This is how you send out results when options are set
# (Note: Mail servers that require authentication are not yet supported.)
## Not run:
y <- cooccurrences("REUTERS", query = "oil")
mail(y)

k <- kwic("REUTERS", query = "oil")
mail(k)

## End(Not run)
```

`means` *calculate means*

Description

calculate means

Usage

```
means(.Object, ...)

## S4 method for signature 'DocumentTermMatrix'
means(.Object, dim = 1)
```

Arguments

`.Object`	object to work on
`...`	further parameters @exportMethod means
`dim`	numeric, 1 or 2 whether to work on rows or columns

ngrams *Get N-Grams*

Description

Count n-grams, either of words, or of characters.

Usage

```
ngrams(.Object, ...)

## S4 method for signature 'partition'
ngrams(.Object, ...)

## S4 method for signature 'character'
ngrams(.Object, ...)

## S4 method for signature 'CorpusOrSubcorpus'
ngrams(.Object, n = 2,
  p_attribute = "word", char = NULL, progress = FALSE, ...)

## S4 method for signature 'partition_bundle'
ngrams(.Object, n = 2, char = NULL,
  p_attribute = "word", mc = FALSE, progress = FALSE, ...)
```

Arguments

.Object	object of class partition
...	further parameters
n	number of tokens/characters
p_attribute	the p-attribute to use (can be > 1)
char	if NULL, tokens will be counted, else characters, keeping only those provided by a character vector
progress	logical
mc	logical, whether to use multicore, passed into call to blapply (see respective documentation)

Examples

```
use("polmineR")
P <- partition("GERMAPARLMINI", date = "2009-10-27")
ngramObject <- ngrams(P, n = 2, p_attribute = "word", char = NULL)

# a more complex scenario: get most frequent ADJA/NN-combinations
ngramObject <- ngrams(P, n = 2, p_attribute = c("word", "pos"), char = NULL)
ngramObject2 <- subset(
```

```
 ngramObject,
 ngramObject[["1_pos"]] == "ADJA"  & ngramObject[["2_pos"]] == "NN"
 )
ngramObject2@stat[, "1_pos" := NULL][, "2_pos" := NULL]
ngramObject3 <- sort(ngramObject2, by = "count")
head(ngramObject3)
```

---

ngrams_class *Ngrams class.*

Description

Ngrams class.

---

noise *detect noise*

Description

detect noise

Usage

```
noise(.Object, ...)

## S4 method for signature 'DocumentTermMatrix'
noise(.Object, minTotal = 2,
  minTfIdfMean = 0.005, sparse = 0.995, stopwordsLanguage = "german",
  minNchar = 2, specialChars = getOption("polmineR.specialChars"),
  numbers = "^[0-9\\.,]+$", verbose = TRUE)

## S4 method for signature 'TermDocumentMatrix'
noise(.Object, ...)

## S4 method for signature 'character'
noise(.Object, stopwordsLanguage = "german",
  minNchar = 2, specialChars = getOption("polmineR.specialChars"),
  numbers = "^[0-9\\.,]+$", verbose = TRUE)

## S4 method for signature 'textstat'
noise(.Object, p_attribute, ...)
```

Arguments

.Object	an .Object of class "DocumentTermMatrix"
...	further parameters
minTotal	minimum colsum (for DocumentTermMatrix) to qualify a term as non-noise
minTfIdfMean	minimum mean value for tf-idf to qualify a term as non-noise
sparse	will be passed into "removeSparseTerms" from "tm"-package
stopwordsLanguage	e.g. "german", to get stopwords defined in the tm package
minNchar	min char length ti qualify a term as non-noise
specialChars	special characters to drop
numbers	regex, to drop numbers
verbose	logical
p_attribute	relevant if applied to a textstat object

Value

a list

---

partition *Initialize a partition.*

Description

Create a subcorpus and keep it in an object of the partition class. If defined, counts are performed for the p-attribute defined by the parameter p_attribute.

Usage

```
partition(.Object, ...)

## S4 method for signature 'character'
partition(.Object, def = NULL, name = "",
  encoding = NULL, p_attribute = NULL, regex = FALSE, xml = "flat",
  decode = TRUE, type = get_type(.Object), mc = FALSE,
  verbose = TRUE, ...)

## S4 method for signature 'environment'
partition(.Object, slots = c("name", "corpus",
  "size", "p_attribute"))

## S4 method for signature 'partition'
partition(.Object, def = NULL, name = "",
  regex = FALSE, p_attribute = NULL, decode = TRUE, xml = NULL,
  verbose = TRUE, mc = FALSE, ...)
```

```
## S4 method for signature 'Corpus'
partition(.Object, def = NULL, name = "",
  encoding = NULL, regex = FALSE, xml = "flat",
  type = get_type(.Object), verbose = TRUE, ...)

## S4 method for signature 'context'
partition(.Object, node = TRUE)
```

Arguments

.Object	A length-one character-vector, the CWB corpus to be used.
...	Arguments to define partition (see examples).
def	A named list of character vectors of s-attribute values, the names are the s-attributes (see details and examples)
name	A name for the new partition object, defaults to "".
encoding	The encoding of the corpus (typically "LATIN1 or "(UTF-8)), if NULL, the encoding provided in the registry file of the corpus (charset="...") will be used.
p_attribute	The p-attribute(s) for which a count is performed.
regex	A logical value (defaults to FALSE).
xml	Either 'flat' (default) or 'nested'.
decode	Logical, whether to turn token ids to strings (set FALSE to minimize object size / memory consumption) in data.table with counts.
type	A length-one character vector specifying the type of corpus / partition (e.g. "plpr")
mc	Whether to use multicore (for counting terms).
verbose	Logical, whether to be verbose.
slots	Object slots that will be reported columns of data.frame summarizing partition objects in environment.
node	A logical value, whether to include the node (i.e. query matches) in the region matrix generated when creating a partition from a context-object.

Details

The function sets up a partition object based on s-attribute values. The s-attributes defining the partition can be passed in as a list, e.g. list(interjection="speech", year = "2013"), or directly (see examples).

The s-attribute values defining the partition may use regular expressions. To use regular expressions, set the parameter regex to TRUE. Regular expressions are passed into grep, i.e. the regex syntax used in R needs to be used (double backlashes etc.). If regex is FALSE, the length of the character vectors can be > 1, matching s-attributes are identifies with the operator '

The XML imported into the CWB may be "flat" or "nested". This needs to be indicated with the parameter xml (default is "flat"). If you generate a partition based on a flat XML structure, some performance gain may be achieved when ordering the s-attributes with decreasingly restrictive

conditions. If you have a nested XML, it is mandatory that the order of the s-attributes provided reflects the hierarchy of the XML: The top-level elements need to be positioned at the beginning of the list with the s-attributes, the the most restrictive elements at the end.

If `p_attribute` is not NULL, a count of tokens in the corpus will be performed and kept in the `stat`-slot of the partition-object. The length of the `p_attribute` character vector may be 1 or more. If two or more p-attributes are provided, The occurrence of combinations will be counted. A typical scenario is to combine the p-attributes "word" or "lemma" and "pos".

If `.Object` is a length-one character vector, a subcorpus/partition for the corpus defined be `.Object` is generated.

If `.Object` is an environment (typically `.GlobalEnv`), the `partition` objects present in the environment are listed.

If `.Object` is a `partition` object, a subcorpus of the subcorpus is generated.

If `.Object` is a `Corpus` object, preparing the `partition` may work more efficiently than if `.Object` is a length-one character vector.

Value

An object of the S4 class `partition`.

Author(s)

Andreas Blaette

See Also

To learn about the methods available for objects of the class `partition`, see `partition_class`,

Examples

```
use("polmineR")
spd <- partition("GERMAPARLMINI", party = "SPD", interjection = "speech")
kauder <- partition("GERMAPARLMINI", speaker = "Volker Kauder", p_attribute = "word")
merkel <- partition("GERMAPARLMINI", speaker = ".*Merkel", p_attribute = "word", regex = TRUE)
s_attributes(merkel, "date")
s_attributes(merkel, "speaker")
merkel <- partition(
  "GERMAPARLMINI", speaker = "Angela Dorothea Merkel",
  date = "2009-11-10", interjection = "speech", p_attribute = "word"
  )
merkel <- subset(merkel, !word %in% punctuation)
merkel <- subset(merkel, !word %in% tm::stopwords("de"))

# a certain defined time segment
days <- seq(
  from = as.Date("2009-10-28"),
  to = as.Date("2009-11-11"),
  by = "1 day"
)
period <- partition("GERMAPARLMINI", date = days)
```

partition_bundle *Generate bundle of partitions.*

Description

Use partition_bundle to create a partition_bundle object, which combines a set of partition objects.

Usage

```
partition_bundle(.Object, ...)

## S4 method for signature 'partition'
partition_bundle(.Object, s_attribute,
  values = NULL, prefix = "", mc = getOption("polmineR.mc"),
  verbose = TRUE, progress = FALSE, type = get_type(.Object), ...)

## S4 method for signature 'character'
partition_bundle(.Object, s_attribute,
  values = NULL, prefix = "", mc = getOption("polmineR.mc"),
  verbose = TRUE, progress = FALSE, xml = "flat",
  type = get_type(.Object), ...)

## S4 method for signature 'context'
partition_bundle(.Object, node = TRUE,
  progress = TRUE, mc = 1L)

## S4 method for signature 'partition_bundle'
partition_bundle(.Object, s_attribute,
  prefix = character(), progress = TRUE,
  mc = getOption("polmineR.mc"))
```

Arguments

.Object	A partition, a length-one character vector supplying a CWB corpus, or a partition_bundle
...	parameters to be passed into partition-method (see respective documentation)
s_attribute	The s-attribute to vary
values	Values the s-attribute provided shall assume.
prefix	A character vector that will be attached as a prefix to partition names.
mc	Logical, whether to use multicore parallelization.
verbose	Logical, whether to provide progress information.
progress	Logical, whether to show progress bar.
type	The type of partition to generate.

xml	logical
node	A logical value, whether to include the node (i.e. query matches) in the region matrix generated when creating a partition from a context-object.

Details

Applying the partition_bundle-method to a partition_bundle-object will iterate through the partition objects in the object-slot in the partition_bundle, and apply partition_bundle on each partition, splitting it up by the s-attribute provided by the argument s_attribute. The return value is a partition_bundle, the names of which will be the names of the incoming partition_bundle concatenated with the s-attribute values used for splitting. The argument prefix can be used to achieve a more descriptive name.

Value

S4 class partition_bundle, with list of partition objects in slot 'objects'

Author(s)

Andreas Blaette

See Also

partition and bundle

Examples

```
use("polmineR")
bt2009 <- partition("GERMAPARLMINI", date = "2009-.*", regex = TRUE)
pb <- partition_bundle(bt2009, s_attribute = "date", progress = TRUE, p_attribute = "word")
dtm <- as.DocumentTermMatrix(pb, col = "count")
summary(pb)
pb <- partition_bundle("GERMAPARLMINI", s_attribute = "date")
# split up objects in partition_bundle by using partition_bundle-method
use("polmineR")
pb <- partition_bundle("GERMAPARLMINI", s_attribute = "date")
pb2 <- partition_bundle(pb, s_attribute = "speaker", progress = FALSE)

summary(pb2)
```

---

partition_bundle-class

Bundle of partitions (partition_bundle class).

Description

Class and methods to manage bundles of partitions.

Usage

```
## S4 method for signature 'partition_bundle'
show(object)

## S4 method for signature 'partition_bundle'
summary(object, progress = TRUE)

## S4 method for signature 'partition_bundle'
merge(x, name = "", verbose = TRUE)

## S4 method for signature 'partition_bundle,ANY,ANY,ANY'
x[i]

## S4 method for signature 'partition_bundle'
barplot(height, ...)

## S4 method for signature 'list'
as.partition_bundle(.Object, ...)

## S4 method for signature 'environment'
partition_bundle(.Object)

## S4 method for signature 'partition_bundle'
enrich(.Object, mc = FALSE,
  progress = TRUE, verbose = FALSE, ...)

## S4 method for signature 'partition_bundle'
s_attributes(.Object, s_attribute, ...)

flatten(object)
```

Arguments

object	a partition_bundle object
progress	logical
x	a partition_bundle object
name	the name for the new partition
verbose	logical
i	integer index
height	height
...	further parameters
.Object	a partition_bundle object
mc	logical or, if numeric, providing the number of cores
s_attribute	the s-attribute to use

Details

The merge-method aggregates several partitions into one partition. The prerequisite for this function to work properly is that there are no overlaps of the different partitions that are to be summarized. Encodings and the root node need to be identical, too.

Using brackets can be used to retrieve the count for a token from the partition objects in a partition_bundle.

Value

An object of the class 'partition. See partition for the details on the class.

a partition_bundle object

Slots

objects Object of class list the partitions making up the bundle

corpus Object of class character the CWB corpus the partition is based on

s_attributes_fixed Object of class list fixed s-attributes

encoding Object of class character encoding of the corpus

explanation Object of class character an explanation of the partition

xml Object of class character whether the xml is flat or nested

call Object of class character the call that generated the partition_bundle

Author(s)

Andreas Blaette

---

partition_class *Partition class and methods.*

Description

The partition class is used to manage subcorpora. It is an S4 class, and a set of methods is defined for the class. The class inherits from the classes count and textstat.

Usage

```
## S4 method for signature 'partition'
summary(object)

## S4 method for signature 'partition'
p_attributes(.Object, p_attribute = NULL, ...)

## S4 method for signature 'partition'
split(x, gap, ...)
```

```
is.partition(x)

## S4 method for signature 'partition'
x$name

## S4 method for signature 'partition,ANY'
e1 == e2

## S4 method for signature 'partition,ANY'
e1 != e2

## S4 method for signature 'partition'
x %in% table

## S4 method for signature 'partition'
zoom(x, ...)

## S4 method for signature 'partition'
enrich(.Object, p_attribute = NULL,
  decode = TRUE, verbose = TRUE, mc = FALSE, ...)

## S4 method for signature 'partition'
as.regions(x)
```

Arguments

object	A partition object.
.Object	A partition object.
p_attribute	a p-attribute (for enriching) / performing count.
...	further parameters passed into count when calling enrich, and ...
x	A partition object.
gap	An integer value specifying the minimum gap between regions for performing the split.
name	An s-attribute that will be assigned as key to a partition.
e1	A first expression, a partition object in this case.
e2	A second expression, the value of an s-attribute.
table	Values a s-attribute shall assume.
decode	logical value, whether to decode token ids into strings when performing count
verbose	logical value, whether to output messages
mc	logical or, if numeric, providing the number of cores

Details

As partition objects inherit from count and textstat class, methods available are view to inspect the table in the stat slot, name and name<- to retrieve/set the name of an object, and more.

The p_attributes-method returns the p-attributes defined for the corpus the partition is derived from, if argument p_attribute is NULL (the default). If p_attribute is defined, the unique values for the p-attribute are returned.

The split-method will split a partition object into a partition_bundle if gap between strucs exceeds a minimum number of tokens specified by gap. Relevant to split up a plenary protocol# into speeches. Note: To speed things up, the returned partitions will not include frequency lists. The lists can be prepared by applying enrich on the partition_bundle object that is returned.

The is.partition function returns a logical value whether x is a partition, or not.

The '$'-method will assign the argument name to the slot key and return the modified object.

The enrich-method will add a count of tokens defined by argument p_attribute to slot stat of the partition object.

Slots

name A name to identify the object (character vector with length 1); useful when multiple partition objects are combined to a partition_bundle.

corpus The CWB indexed corpus the partition is derived from (character vector with length 1).

encoding Encoding of the corpus (character vector with length 1).

s_attributes A named list with the s-attributes specifying the partition.

explanation Object of class character, an explanation of the partition.

cpos A matrix with left and right corpus positions defining regions (two columns).

annotations Object of class list.

size Total size of the partition (integer vector, length 1).

stat An (optional) data.table with counts. If present, speeds up computation of cooccurrences, as count is already present.

metadata Object of class data.frame, metadata information.

strucs Object of class intger, the strucs defining the partition.

p_attribute Object of class character indicating the p_attribute of the count in slot stat.

xml Object of class character, whether the xml is flat or nested.

s_attribute_strucs Object of class character the base node

key Experimental, an s-attribute that is used as a key.

call Object of class character the call that generated the partition

Author(s)

Andreas Blaette

See Also

The partition-class inherits from the textstat-class, see respective documentation to learn more.

Examples

```
m <- partition("GERMAPARLMINI", speaker = "Merkel", regex = TRUE)
m2 <- m$speaker
m2@key
m <- partition("GERMAPARLMINI", speaker = "Merkel", regex = TRUE)
m$date == "2009-10-28"
s <- partition("GERMAPARLMINI", interjection = "speech")
s$party != "NA"
s <- partition("GERMAPARLMINI", interjection = "speech")
s$date %in% c("2009-10-27", "2009-10-28")
m <- partition("GERMAPARLMINI", speaker = "Merkel", regex = TRUE)
y <- zoom(m, date == "2009-10-28")

speeches <- partition("GERMAPARLMINI", interjection = "speech")
m <- zoom(speeches, date == "2009-10-28" & speaker == "Angela Dorothea Merkel")

not_unknown <- zoom(speeches, party != c("NA", "FDP"))
s_attributes(not_unknown, "party")
```

---

pmi *Calculate Pointwise Mutual Information (PMI).*

Description

Pointwise mutual information (PMI) is calculated as follows, see Manning/Schuetze (1999) for an explanation:

$$I(x, y) = log \frac{p(x, y)}{p(x)p(y)}$$

Note that the computation uses log base 2, not the natural logarithm you find in examples (e.g. https://en.wikipedia.org/wiki/Pointwise_mutual_information).

Usage

```
pmi(.Object)

## S4 method for signature 'context'
pmi(.Object)
```

Arguments

.Object	An object.

References

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 178-183.

See Also

See ll, chisquare and t_test.

Examples

```
y <- cooccurrences("REUTERS", query = "oil", method = "pmi")
N <- size(y)[["partition"]]
I <- log2((y[["count_coi"]]/N) / ((count(y) / N) * (y[["count_partition"]] / N)))
```

---

p_attributes *Get p-attributes.*

Description

In a CWB corpus, every token has positional attributes. While s-attributes cover a range of tokens, every single token in the token stream of a corpus will have a set of positional attributes (such as part-of-speech, or lemma). The available p-attributes are returned by the p_attributes-method.

Usage

```
p_attributes(.Object, ...)

## S4 method for signature 'character'
p_attributes(.Object, p_attribute = NULL, ...)
```

Arguments

.Object	a character vector (length 1) or partition object
...	further arguments
p_attribute	p-attribute to decode

References

Stefan Evert & The OCWB Development Team, CQP Query Language Tutorial, http://cwb.sourceforge.net/files/CQP_Tutoria

Examples

```
use("polmineR")
p_attributes("GERMAPARLMINI")
```

read *Display full text.*

Description

Generate text (i.e. html) and display it in the viewer pane of RStudio for reading it. If called on a partition_bundle-object, skip through the partitions contained in the bundle.

Usage

```
read(.Object, ...)

## S4 method for signature 'partition'
read(.Object, meta = NULL, highlight = list(),
  tooltips = list(), verbose = TRUE, cpos = TRUE,
  cutoff = getOption("polmineR.cutoff"),
  template = get_template(.Object), ...)

## S4 method for signature 'partition_bundle'
read(.Object, highlight = list(),
  cpos = TRUE, ...)

## S4 method for signature 'data.table'
read(.Object, col, partition_bundle,
  highlight = list(), cpos = FALSE, ...)

## S4 method for signature 'hits'
read(.Object, def, i = NULL, ...)

## S4 method for signature 'kwic'
read(.Object, i = NULL,
  type = registry_get_properties(corpus(.Object))["type"])

## S4 method for signature 'regions'
read(.Object, meta = NULL)
```

Arguments

.Object	an object to be read ("partition" or "partition_bundle")
...	further parameters passed into read
meta	a character vector supplying s-attributes for the metainformation to be printed; if not stated explicitly, session settings will be used
highlight	a named list of character vectors (see details)
tooltips	a named list (names are colors, vectors are tooltips)
verbose	logical

cpos	logical, if TRUE, corpus positions will be assigned (invisibly) to a cpos tag of a html element surrounding the tokens
cutoff	maximum number of tokens to display
template	template to format output
col	column of data.table with terms to be highlighted
partition_bundle	a partition_bundle object
def	a named list used to define a partition (names are s-attributes, vectors are values of s-attributes)
i	if .Object is an object of the classes kwic or hits, the ith kwic line or hit to derive a partition to be inspected from
type	the partition type, see documentation for partition-method

Details

To prepare the html output, the method read will call html and as.markdown subsequently, the latter method being the actual worker. Consult these methods to understand how preparing the output works.

The param highlight can be used to highlight terms. It is expected to be a named list of character vectors, the names providing the colors, and the vectors the terms to be highlighted. To add tooltips, use the param tooltips.

The method read is a high-level function that calls the methods mentioned before. Results obtained through read can also be obtained through combining these methods in a pipe using the package magrittr. That may offer more flexibility, e.g. to highlight matches for CQP queries. See examples and the documentation for the different methods to learn more.

See Also

For concordances / a keword-in-context display, see kwic.

Examples

```
use("polmineR")
merkel <- partition("GERMAPARLMINI", date = "2009-11-10", speaker = "Merkel", regex = TRUE)
read(merkel, meta = c("speaker", "date"))
read(
  merkel,
  highlight = list(yellow = c("Deutschland", "Bundesrepublik"), lightgreen = "Regierung"),
  meta = c("speaker", "date")
)

## Not run:
pb <- as.speeches("GERMAPARLMINI", s_attribute_date = "date", s_attribute_name = "speaker")
pb <- pb[[ data.table::as.data.table(summary(pb))[size >= 500][["name"]] ]]
pb <- pb[[ 1:10 ]]
read(pb)

## End(Not run)
```

regions *Regions of a CWB corpus.*

Description

A coerce-method is available to coerce a partition object to a regions object.

Usage

```
as.regions(x, ...)

## S3 method for class 'regions'
as.data.table(x, values = NULL)
```

Arguments

x	object of class regions
...	Further arguments.
values	values to assign to a column that will be added

Details

The virtual class CorpusOrSubcorpus is a way to handle corpora specified by a character vector, region objects, and partition objects in a uniform manner.

The as.regions-method coerces objects to a regions-object.

Slots

cpos a two-column data.table that will include a "cpos_left" and "cpos_right" column

corpus the CWB corpus (character vector length 1)

encoding the encoding of the CWB corpus (character vector length 1)

See Also

Other classes to manage corpora: corpus-class, subcorpus-class

Examples

```
use("polmineR")
P <- partition("GERMAPARLMINI", date = "2009-11-12", speaker = "Jens Spahn")
R <- as.regions(P)
```

registry *Get registry and data directories.*

Description

The Corpus Workbench (CWB) uses a registry directory with (txt) files describing corpora in a standardized format. The binary files of a corpus are stored in a data directory defined in the registry directory. The `registry` and `data_dir` functions return the respective direcories within a package, if the argument `pkg` is used, or the temporary registry and data directory in the per-session temporary directory, if `pkg` is `NULL` (default value).

Usage

```
registry(pkg = NULL)

data_dir(pkg = NULL)
```

Arguments

`pkg`	A character string with the name of a single package; if `NULL` (default), the temporary registry and data directory is returned.

Details

Upon loading the polmineR package, there is a check whether the environment variable `CORPUS_REGISTRY` is defined. In case it is, the registry files in the directory defined by the `CORPUS_REGISTRY` environment variable are copied to the temporary registry directory, which serves as the central place to store all registry files for all corpora, be it system corpora, corpora included in R packages, or temporary corpora.

Value

A path to a (registry or data) directory, or NULL, if package does not exist or is not a package including a corpus.

Examples

```
registry() # returns temporary registry directory
registry(pkg = "polmineR") # returns registry directory in polmineR-package

data_dir()
data_dir(pkg = "polmineR")
```

registry_get_name *Evaluate registry file.*

Description

Functions to extract information from a registry file describing a corpus. Several operations could be accomplished with the 'cwb-regedit' tool, the functions defined here ensure that manipulating the registry is possible without a full installation of the CWB.

Usage

```
registry_get_name(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_id(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_home(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_info(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_encoding(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_p_attributes(corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_s_attributes(corpus,
  registry = Sys.getenv("CORPUS_REGISTRY"))

registry_get_properties(corpus, registry = Sys.getenv("CORPUS_REGISTRY"))
```

Arguments

corpus	name of the CWB corpus
registry	directory of the registry (defaults to CORPUS_Registry environment variable)

Details

An appendix to the 'Corpus Encoding Tutorial' (http://cwb.sourceforge.net/files/CWB_Encoding_Tutorial.pdf) includes an explanation of the registry file format.

registry_reset *Reset registry directory.*

Description

A utility function to reset the environment variable CORPUS_REGISTRY. That may be necessary if you want use a CWB corpus that is not stored in the usual place. In particular, resetting the environment variable is required if you want to use a corpus delivered in a R package,

Usage

```
registry_reset(registryDir = registry(), verbose = TRUE)
```

Arguments

registryDir	path to the registry directory to be used
verbose	logical, whether to be verbose

Details

Resetting the CORPUS_REGISTRY environment variable is also necessary for the interface to CWB corpora.

To get the path to a package that contains a CWB corpus, use `system.file` (see examples).

Value

the registry directory used before resetting CORPUS_REGISTRY

See Also

To conveniently reset registry, see use.

Examples

```
x <- system.file(package = "polmineR", "extdata", "cwb", "registry")
registry_reset(registryDir = x)
```

renamed *Renamed Functions*

Description

These functions have been renamed in order to have a consistent coding style that follows the snake_case convention. The "old" function still work to maintain backwards compatiblity.

Usage

```
sAttributes(...)

pAttributes(...)

getTokenStream(...)

getTerms(...)

getEncoding(...)

partitionBundle(...)

as.partitionBundle(...)

setTemplate(...)

getTemplate(...)
```

Arguments

...	argument that are passed to the renamed function

size *Get Number of Tokens.*

Description

The method will get the number of tokens in a corpus or partition, or the dispersion across one or more s-attributes.

Usage

```
size(x, ...)

## S4 method for signature 'character'
size(x, s_attribute = NULL, verbose = TRUE, ...)

## S4 method for signature 'partition'
size(x, s_attribute = NULL, ...)

## S4 method for signature 'DocumentTermMatrix'
size(x)

## S4 method for signature 'TermDocumentMatrix'
size(x)

## S4 method for signature 'features'
size(x)
```

Arguments

x	object to get size(s) for
...	further arguments
s_attribute	character vector with s-attributes (one or more)
verbose	logical, whether to print messages

Details

One or more s-attributes can be provided to get the dispersion of tokens across one or more dimensions. Two or more s-attributes can lead to reasonable results only if the corpus XML is flat.

The size-method for features objects will return a named list with the size of the corpus of interest ("coi"), i.e. the number of tokens in the window, and the reference corpus ("ref"), i.e. the number of tokens that are not matched by the query and that are outside the window.

Value

an integer vector if s_attribute is NULL, a data.table otherweise

See Also

See dispersion-method for counts of hits. The hits method calls the size-method to get sizes of subcorpora.

Examples

```
use("polmineR")
size("GERMAPARLMINI")
size("GERMAPARLMINI", s_attribute = "date")
size("GERMAPARLMINI", s_attribute = c("date", "party"))
```

```
P <- partition("GERMAPARLMINI", date = "2009-11-11")
size(P, s_attribute = "speaker")
size(P, s_attribute = "party")
size(P, s_attribute = c("speaker", "party"))
```

---

slice *Virtual class clice.*

Description

The classes regions and partition can be used to define subcorpora. Unlike the regions class, the partition class may include statistical evaluations. The virtual class slice is a mechanism to define methods for these classes without making regions the superclass of partition.

Usage

```
## S4 method for signature 'slice'
aggregate(x)
```

Arguments

x	An object of a class belonging to the virtual class slice, i.e. a partition or regions object.

Details

The method aggregate will deflate the matrix in the slot cpos, i.e. it checks for each new row in the matrix whether it increments the end of the previous region (by 1), and ensure that the cpos matrix defines disjoined regions.

Examples

```
P <- new(
  "partition",
  cpos = matrix(data = c(1:10, 20:29), ncol = 2, byrow = TRUE),
  stat = data.table::data.table()
)
P2 <- aggregate(P)
P2@cpos
```

store *Store objects as Excel-file.*

Description

Store objects as Excel-file.

Usage

```
store(.Object, ...)

## S4 method for signature 'textstat'
store(.Object, filename = tempfile(fileext =
  ".xlsx"), rows = 1L:nrow(.Object))

## S4 method for signature 'data.frame'
store(.Object, filename = tempfile(fileext =
  ".xlsx"), rows = 1L:nrow(.Object))

## S4 method for signature 'kwic'
store(.Object, filename = tempfile(fileext = ".xlsx"),
  rows = 1L:nrow(.Object))
```

Arguments

.Object	An object that can be processed.
...	Further arguments.
filename	Name of the file to write.
rows	The rows of the table to export.

subcorpus-class *S4 subcorpus class.*

Description

S4 subcorpus class.

Usage

```
## S4 method for signature 'subcorpus'
show(object)
```

Arguments

object	A subcorpus object.

Methods (by generic)

- show: Output basic information about subcorpus object.

Slots

s_attributes A named list with the structural attributes defining the subcorpus.

cpos A matrix with left and right corpus positions defining regions (two columns).

annotations Object of class list.

size Total size of the subcorpus (length-one integer vector).

metadata Object of class data.frame, metadata information.

strucs Object of class integer, the strucs defining the subcorpus.

xml Object of class character, whether the xml is "flat" or "nested".

s_attribute_strucs Object of class character, the base node.

key Experimental, an s-attribute that is used as a key.

See Also

Other classes to manage corpora: corpus-class, regions

---

s_attributes *Get s-attributes.*

Description

Structural annotations (s-attributes) of a corpus provide metainformation for regions of tokens. Gain access to the s-attributes available for a corpus or partition, or the values of s-attributes in a corpus/partition with the s_attributes-method.

Usage

```
s_attributes(.Object, ...)

## S4 method for signature 'character'
s_attributes(.Object, s_attribute = NULL,
  unique = TRUE, regex = NULL, ...)

## S4 method for signature 'partition'
s_attributes(.Object, s_attribute = NULL,
  unique = TRUE, ...)
```

Arguments

.Object	either a partition object or a character vector specifying a CWB corpus
...	to maintain backward compatibility, of argument sAttribute is used
s_attribute	name of a specific s-attribute
unique	logical, whether to return unique values only
regex	filter return value by applying a regex

Details

Importing XML into the Corpus Workbench (CWB) turns elements and element attributes into so-called s-attributes. There are two uses of the s_attributes-method: If the s_attribute parameter is NULL (default), the return value is a character vector with all s-attributes present in a corpus.

If s_attribute is the name of a specific s-attribute (a length 1 character vector), the values of the s-attributes available in the corpus/partition are returned.

If a character vector of s-attributes is provided, the method will return a data.table.

Value

a character vector

Examples

```
use("polmineR")

s_attributes("GERMAPARLMINI")
s_attributes("GERMAPARLMINI", "date") # dates of plenary meetings

P <- partition("GERMAPARLMINI", date = "2009-11-10")
s_attributes(P)
s_attributes(P, "speaker") # get names of speakers
```

---

tempcorpus *create a tempcorpus*

Description

Based on the corpus positions defining a partition, a temporary CWB corpus is generated that is stored in a temporary directory.

Usage

```
tempcorpus(.Object, ...)
```

Arguments

.Object	a partition object
...	further parameters

tempcorpus_class *S4 class to capture core information on a temporary CWB corpus*

Description

S4 class to capture core information on a temporary CWB corpus

Slots

`cpos` matrix with start/end corpus positions

`dir` directory where the tempcorpus is stored

`registry` directory of the registry dir (subdirectory of dir)

`indexed` directory of the dir with the indexed files

terms *Get terms in* `partition` *or corpus.*

Description

Get terms in `partition` or corpus.

Usage

```
## S4 method for signature 'partition'
terms(x, p_attribute, regex = NULL, ...)

## S4 method for signature 'character'
terms(x, p_attribute, regex = NULL,
  robust = FALSE, ...)
```

Arguments

`x`	an atomic `character` vector with a corpus id or `partition` object
`p_attribute`	the p-attribute to be analyzed
`regex`	regular expression(s) to filter results
`...`	for backward compatibility
`robust`	logical, whether to check for potential failures

Examples

```
use("polmineR")
session <- partition("GERMAPARLMINI", date = "2009-10-27")
words <- terms(session, "word")
terms(session, p_attribute = "word", regex = "^Arbeit.*")
terms(session, p_attribute = "word", regex = c("Arbeit.*", ".*arbeit"))

terms("GERMAPARLMINI", p_attribute = "word")
terms("GERMAPARLMINI", p_attribute = "word", regex = "^Arbeit.*")
```

---

textstat-class *S4 textstat superclass.*

Description

The textstat-class (technically an S4 class) serves as a superclass for the classes features, context, and partition. Usually, the class will not be used directly. It offers a set of standard generic methods (such as head, tail, dim, nrow, colnames) its childs inherit. The core feature of textstat and its childs is a data.table in the slot stat for keeping data on text statistics of a corpus, or a partition.

Usage

```
## S4 method for signature 'textstat'
name(x)

## S4 method for signature 'character'
name(x)

## S4 replacement method for signature 'textstat,character'
name(x) <- value

## S4 method for signature 'textstat'
round(x, digits = 2L)

## S4 method for signature 'textstat'
sort(x, by, decreasing = TRUE)

as.bundle(object, ...)

## S4 method for signature 'textstat,textstat'
e1 + e2

## S4 method for signature 'textstat'
subset(x, ...)

## S3 method for class 'textstat'
```

```
as.data.table(x)

## S4 method for signature 'textstat'
show(object)

## S4 method for signature 'textstat'
p_attributes(.Object)

## S4 method for signature 'textstat'
knit_print(x, options = knitr::opts_chunk, ...)

## S4 method for signature 'textstat'
format(x, digits = 2L)

## S4 method for signature 'textstat'
view(.Object)
```

Arguments

x	A textstat object.
value	A character vector to assign as name to slot name of a textstat class object.
digits	Number of digits.
by	Column that will serve as the key for sorting.
decreasing	Logical, whether to return decreasing order.
object	a textstat object
...	Further arguments.
e1	A texstat object.
e2	Another texstat object.
.Object	A textstat object.
options	Chunk options.

Details

A head-method will return the first rows of the data.table in the stat-slot. Use argument n to specify the number of rows.

A tail-method will return the last rows of the data.table in the stat-slot. Use argument n to specify the number of rows.

The methods dim, nrow and ncol will return information on the dimensions, the number of rows, or the number of columns of the data.table in the stat-slot, respectively.

Objects derived from the textstat class can be indexed with simple square brackets ("[") to get rows specified by an numeric/integer vector, and with double square brackets ("[[") to get specific columns from the data.table in the slot stat.

The colnames-method will return the column names of the data-table in the slot stat.

The methods as.data.table, and as.data.frame will extract the data.table in the slot stat as a data.table, or data.frame, respectively.

textstat objects can have a name, which can be retrieved, and set using the name-method and name<-, respectively.

The round()-method looks up all numeric columns in the data.table in the stat-slot of the textstat object and rounds values of these columns to the number of decimal places specified by argument digits.

The format()-method returns a pretty-printed and minimized version of the data.table in the stat-slot of the textstat-object: It will round all numeric columns to the number of decimal numbers specified by digits, and drop all columns with token ids. The return value is a data.table.

Slots

p_attribute Object of class character, p-attribute of the query.

corpus A corpus specified by a length-one character vector.

stat A data.table with statistical information.

name The name of the object.

encoding A length-one character vector, the encoding of the corpus.

Examples

```
use("polmineR")
P <- partition("GERMAPARLMINI", date = ".*", p_attribute = "word", regex = TRUE)
y <- cooccurrences(P, query = "Arbeit")

# Standard generic methods known from data.frames work for objects inheriting
# from the textstat class

head(y)
tail(y)
nrow(y)
ncol(y)
dim(y)
colnames(y)

# Use brackets for indexing

## Not run:
y[1:25]
y[,c("word", "ll")]
y[1:25, "word"]
y[1:25][["word"]]
y[which(y[["word"]] %in% c("Arbeit", "Sozial"))]
y[ y[["word"]] %in% c("Arbeit", "Sozial") ]

## End(Not run)
```

tooltips *Add tooltips to text output.*

Description

Highlight tokens based on exact match, a regular expression or corpus position in kwic output or html document.

Usage

```
tooltips(.Object, tooltips, ...)

## S4 method for signature 'character'
tooltips(.Object, tooltips = list())

## S4 method for signature 'html'
tooltips(.Object, tooltips = list())

## S4 method for signature 'kwic'
tooltips(.Object, tooltips, regex = FALSE, ...)
```

Arguments

.Object	A html or character object with html.
tooltips	A named list of character vectors, the names need to match colors in the list provided to param highlight. The value of the character vector is the tooltip to be displayed.
...	Further arguments are interpreted as assignments of tooltips to tokens.
regex	Logical, whether character vector values of argument tooltips are interpreted as regular expressions.

Examples

```
use("polmineR")

P <- partition("REUTERS", places = "argentina")
H <- html(P)
Y <- highlight(H, lightgreen = "higher")
T <- tooltips(Y, list(lightgreen = "Further information"))
if (interactive()) T

# Using the tooltips-method in a pipe ...
if (require("magrittr")){
  P %>%
    html() %>%
    highlight(yellow = c("barrels", "oil", "gas")) %>%
    tooltips(list(yellow = "energy"))
}
```

`trim` *trim an object*

Description

Method to trim and adjust objects by applying thresholds, minimum frequencies etc. It can be applied to `context`, `features`, `context`, `partition` and `partition_bundle` objects.

Usage

```
trim(object, ...)

## S4 method for signature 'TermDocumentMatrix'
trim(object, termsToKeep = NULL,
  termsToDrop = NULL, docsToKeep = NULL, docsToDrop = NULL,
  verbose = TRUE)

## S4 method for signature 'DocumentTermMatrix'
trim(object, ...)

punctuation
```

Arguments

`object`	the object to be trimmed
`...`	further arguments
`termsToKeep`	...
`termsToDrop`	...
`docsToKeep`	...
`docsToDrop`	...
`verbose`	logical

Format

An object of class `character` of length 13.

Author(s)

Andreas Blaette

t_test *Perform t-test.*

Description

S4 method to perform t-test.

Usage

```
t_test(.Object)

## S4 method for signature 'context'
t_test(.Object)
```

Arguments

.Object	A context or features object

Details

The calculation of the t-test is based on the formula

$$t = \frac{\overline{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$

where μ is the mean of the distribution, x the sample mean, s^2 the sample variance, and N the sample size.

References

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 163-166.

use *Add corpora in R data packages to session registry.*

Description

Use CWB indexed corpora in R data packages by adding registry file to session registry.

Usage

```
use(pkg, lib.loc = .libPaths(), tmp = FALSE, verbose = TRUE)
```

Arguments

pkg	A package including at least one CWB indexed corpus.
lib.loc	A character vector with path names of R libraries.
tmp	Whether to use a temporary data directory.
verbose	Logical, whether to output status messages.

Details

pkg is expected to be an installed data package that includes CWB indexed corpora. The use-function will add the registry files describing the corpus (or the corpora) to the session registry directory and adjust the path pointing to the data in the package.

The registry files within the package are assumed to be in the subdirectory `./extdata/cwb/registry` of the installed package. The data directories for corpora are assumed to be in a subdirectory named after the corpus (lower case) in the package subdirectory `./extdata/cwb/indexed_corpora/`. When adding a corpus to the registry, templates for formatting fulltext output are reloaded.

If the path to the data directory in a package includes a non-ASCII character, binary data files of the corpora in package are copied to a subdirectory of the per-session temporary data directory.

See Also

To get the session registry directory, see `registry`; to reset the registry, see `registry_reset`.

Examples

```
use("polmineR")
corpus()
```

---

view *Inspect object using View().*

Description

Inspect object using View().

Usage

```
view(.Object, ...)
```

Arguments

.Object	an object
...	further parameters

weigh *Apply Weight to Matrix*

Description

Apply Weight to Matrix

Usage

```
weigh(.Object, ...)

## S4 method for signature 'TermDocumentMatrix'
weigh(.Object, method = "tfidf")

## S4 method for signature 'DocumentTermMatrix'
weigh(.Object, method = "tfidf")

## S4 method for signature 'count'
weigh(.Object, with)

## S4 method for signature 'count_bundle'
weigh(.Object, with, progress = TRUE)
```

Arguments

.Object	A matrix, or a count-object.
...	further parameters
method	The kind of weight to apply.
with	A data.table used to weigh p-attributes. A column 'weight' with term weights is required, and columns with the p-attributes of .Object for matching.
progress	Logical, whether to show a progress bar.

Examples

```
## Not run:
library(data.table)
if (require("zoo") && require("devtools") && require("magrittr")){

# Source in function 'get_sentiws' from a GitHub gist
gist_url <- file.path(
  "gist.githubusercontent.com",
  "PolMine",
  "70eeb095328070c18bd00ee087272adf",
  "raw",
  "c2eee2f48b11e6d893c19089b444f25b452d2adb",
  "sentiws.R"
 )
```

```
devtools::source_url(sprintf("https://%s", gist_url))
SentiWS <- get_sentiws()

# Do the statistical word context analysis
use("GermaParl")
options("polmineR.left" = 10L)
options("polmineR.right" = 10L)
df <- context("GERMAPARL", query = "Islam", p_attribute = c("word", "pos")) %>%
  partition_bundle(node = FALSE) %>%
  set_names(s_attributes(., s_attribute = "date")) %>%
  weigh(with = SentiWS) %>%
  summary()

# Aggregate by year
df[["year"]] <- as.Date(df[["name"]]) %>% format("%Y-01-01")
df_year <- aggregate(df[,c("size", "positive_n", "negative_n")], list(df[["year"]]), sum)
colnames(df_year)[1] <- "year"

# Use shares instead of absolute counts
df_year$negative_share <- df_year$negative_n / df_year$size
df_year$positive_share <- df_year$positive_n / df_year$size

# Turn it into zoo object, and plot it
Z <- zoo(
  x = df_year[, c("positive_share", "negative_share")],
  order.by = as.Date(df_year[,"year"])
)
plot(
  Z, ylab = "polarity", xlab = "year",
  main = "Word context of 'Islam': Share of positive/negative vocabulary",
  cex = 0.8,
  cex.main = 0.8
)

# Note that we can uses the kwic-method to check for the validity of our findings
words_positive <- SentiWS[weight > 0][["word"]]
words_negative <- SentiWS[weight < 0][["word"]]
kwic("GERMAPARL", query = "Islam", positivelist = c(words_positive, words_negative)) %>%
  highlight(lightgreen = words_positive, orange = words_negative) %>%
  tooltips(setNames(SentiWS[["word"]], SentiWS[["weight"]]))

}

## End(Not run)
```

Index