

# Package ‘polmineR’

October 14, 2022

**Type** Package

**Title** Verbs and Nouns for Corpus Analysis

**Version** 0.8.7

**Date** 2022-08-26

**Depends** R (>= 3.5.0)

**Imports** methods, data.table (>= 1.12.2), fs, slam, Matrix, tm, DT, xml2, stringi, utils, jsonlite, parallel, pbapply, RcppCWB (>= 0.5.3), magrittr, knitr, lifecycle, rlang

**Suggests** markdown, rmarkdown, htmltools (>= 0.4.0), highlight, sendmailR, shiny, shinythemes, shinyjs, miniUI, rhandsontable, testthat, tidytext, covr, igraph, NLP, httr, protolite, curl

**VignetteBuilder** knitr

**Description** Package for corpus analysis using the Corpus Workbench ('CWB', <<https://cwb.sourceforge.io>>) as an efficient back end for indexing and querying large corpora. The package offers functionality to flexibly create subcorpora 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://github.com/PolMine/polmineR>

**Encoding** UTF-8

**Collate** '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' 'annotations.R' 'as.VCorpus.R' 'as.markdown.R' 'kwic.R' 'decode.R' 'cooccurrences.R'

'as.sparseMatrix.R' 'as.speeches.R' 'blapply.R' 'coerce.R'  
 'hits.R' 'cpos.R' 'dispersion.R' 'dotplot.R' 'encoding.R'  
 'enrich.R' 'format.R' 'highlight.R' 'html.R' 'info.R' 'means.R'  
 'noise.R' 'opencpu.R' 'phrases.R' 'polmineR-defunct.R'  
 'ranges.R' 'regions.R' 'read.R' 'registry.R' 'reindex.R'  
 'renamed.R' 'restore.R' 's\_attributes.R' 'size.R' 'split.R'  
 'stats.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** 7.2.0

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Andreas Blaette [aut, cre] (<<https://orcid.org/0000-0001-8970-8010>>),  
 Christoph Leonhardt [ctb],  
 Marius Bertram [ctb]

**Maintainer** Andreas Blaette <[andreas.blaette@uni-due.de](mailto:andreas.blaette@uni-due.de)>

**Repository** CRAN

**Date/Publication** 2022-08-26 15:50:02 UTC

## R topics documented:

polmineR-package	4
annotations	6
as.markdown	8
as.sparseMatrix	10
as.speeches	11
as.TermDocumentMatrix	13
as.VCorpus	16
blapply	16
bundle-class	17
chisquare	20
context	22
context-class	27
context_bundle-class	29
cooccurrences	30
Cooccurrences,corpus-method	35
Cooccurrences-class	39
cooccurrences-class	42
corpus-class	43
corpus-methods	45
count	47
count_class	51
cpos	53
cqp	56
decode	57
dispersion	61
dotplot	64

encoding . . . . .	65
encodings . . . . .	66
enrich . . . . .	67
features . . . . .	68
features-class . . . . .	70
get_template . . . . .	71
get_token_stream . . . . .	72
get_type . . . . .	76
highlight . . . . .	77
hits . . . . .	79
hits_class . . . . .	82
html . . . . .	83
kwic . . . . .	85
kwic-class . . . . .	91
ll . . . . .	95
means . . . . .	98
ngrams . . . . .	99
ngrams_class . . . . .	101
noise . . . . .	101
ocpu_exec . . . . .	103
partition . . . . .	104
partition_bundle . . . . .	107
partition_bundle-class . . . . .	109
partition_class . . . . .	111
partition_to_string . . . . .	114
phrases . . . . .	114
pmi . . . . .	116
polmineR-defunct . . . . .	118
polmineR-generics . . . . .	118
p_attributes . . . . .	119
ranges . . . . .	120
ranges-class . . . . .	122
read . . . . .	122
regions . . . . .	125
registry_move . . . . .	126
renamed . . . . .	128
size . . . . .	128
slice . . . . .	130
subcorpus . . . . .	131
subcorpus_bundle-class . . . . .	133
subset . . . . .	135
s_attributes . . . . .	137
terms . . . . .	139
textstat-class . . . . .	140
tooltips . . . . .	145
trim . . . . .	146
t_test . . . . .	147
use . . . . .	148

view . . . . .	149
weigh . . . . .	150

<b>Index</b>	<b>152</b>
--------------	------------

---

polmineR-package	<i>polmineR-package</i>
------------------	-------------------------

---

## 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()`.

## Package options

- *polmineR.p\_attribute*: The default positional attribute
- *polmineR.left*: Default value for left context.
- *polmineR.lineview*: A logical value, whether ...
- *polmineR.pagelength*: 10L
- *polmineR.meta*:
- *polmineR.mc*:
- *polmineR.cores*:
- *polmineR.browse*:
- *polmineR.buttons*:
- *polmineR.specialChars*:
- *polmineR.cutoff*:

- *polmineR.mdsb*: A list of pairs of character vectors defining regular expression substitutions applied as part of preprocessing documents for html display. Intended usage: Remove characters that would be misinterpreted as markdown formatting instructions.
- *polmineR.corpus\_registry*: The system corpus registry directory defined by the environment variable CORPUS\_REGISTRY before the polmineR package has been loaded. The polmineR package uses a temporary registry directory to be able to use corpora stored at multiple locations in one session. The path to the system corpus registry directory captures this setting to keep it available if necessary.
- *polmineR.shiny*: A logical value, whether polmineR is used in the context of a shiny app. Used to control the appearance of progress bars depending on whether shiny app is running, or not.
- *polmineR.warn.size*: When generating HTML table widgets (e.g. when preparing kwic output to be displayed in RStudio's View pane), the function DT::datatable() that is used internally will issue a warning by default if the object size of the table is greater than 1500000. The warning addresses a client-server scenario that is not applicable in the context of a local RStudio session, so you may want to turn it off. Internally, the warning can be suppressed by setting the option DT.warn.size to FALSE. The polmineR option polmineR.warn.size is processed by functions calling DT::datatable() to set and reset the value of DT.warn.size. Please note: The formulation of the warning does not match the scenario of a local RStudio session, but it may still be useful to get a warning when tables are large and slow to process. Therefore, the default value of the setting is FALSE.

### 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
use(pkg = "RcppCWB", corpus = "REUTERS")

# The package includes two sample corpora
corpus("REUTERS") %>% show_info()
corpus("GERMAPARLMINI") %>% show_info()

# 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)
if (interactive()) read(p)
h <- html(p)
h_highlighted <- highlight(h, highlight = list(yellow = "oil"))
if (interactive()) 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")

```

---

 annotations

*Annotation functionality*


---

## Description

Objects that contain analytical results (kwic objects, objects inheriting from the `textstat` class) can be annotated by creating an annotation layer using the `annotations`-method. The augmented object can be annotated using a shiny gadget by invoking the `edit`-method on it. Note that operations are deliberately in-place, to prevent an unwanted loss of work.

## Usage

```

annotations(x, ...)

## S4 method for signature 'kwic'
annotations(x, i, j, value)

## S4 method for signature 'textstat'
annotations(x, i, j, value)

annotations(x) <- value

## S4 replacement method for signature 'kwic,list'
annotations(x) <- value

```

```
## S4 replacement method for signature 'textstat,list'
annotations(x) <- value

## S4 method for signature 'textstat'
edit(name, viewer = shiny::paneViewer(minHeight = 550), ...)
```

### Arguments

x	An object to be annotated, a kwic class object, or an object inheriting from the textstat class.
...	Passed into rhandsontable::rhandsontable, can be used for settings such as height etc.
i	The row number (single integer value) of the data.table where a new value shall be assigned.
j	The column number (single integer value) of the data.table where a new value shall be assigned.
value	A value to assign.
name	An S4 object to be annotated.
viewer	The viewer to use, see <a href="#">viewer</a> .

### Details

The edit-method is designed to be used in a RStudio session. It generates a shiny gadget (see <https://shiny.rstudio.com/articles/gadgets.html>) shown in the viewer pane of RStudio.

The edit-method returns the modified input object. Note however that changes of annotations are deliberately in-place operations: The input object is changed even if you do not close the gadget "properly" by hitting the "Done" button and catch the modified object. That may be forgotten easily and would be painful after the work that may have been invested.

Consult the examples for the intended workflow.

### Value

The modified input object is returned invisibly.

### Examples

```
use(pkg = "RcppCWB", corpus = "REUTERS")

a <- 2
# upon initializing a kwic object, there is a minimal labels object
# in the labels slot of the kwic object, which we can get using the
# annotations-method
o <- kwic("REUTERS", query = "oil")
annotations(o) # see the result (a data.table)

# assign new annotations as follows, using the reference semantics of the
# data.table you get by calling the labels-method on an object
```

```

annotations(o) <- list(name = "class", what = factor(x = "a", levels = c("a", "b", "c")))
annotations(o) <- list(name = "description", what = "")
annotations(o) # inspect the result

# assign values; note that is an in-place operation using the reference
# semantics of the data.table
# annotations(o, i = 77, j = 1, value = FALSE)
# annotations(o, i = 78, j = 1, value = FALSE)
annotations(o)

## Not run:
edit(o)
annotations(o) # to see changes made

# maybe we want additional metadata
enrich(o, s_attributes = "places")
edit(o)
annotations(o)

# to get some extra context
o <- enrich(o, extra = 5L, table = TRUE)
edit(o)

# lineview may be better when you use a lot of extra context
options(polmineR.lineview = TRUE)
o <- kwic("REUTERS", "oil")
o <- enrich(o, extra = 20L)
edit(o)

x <- cooccurrences("REUTERS", query = "oil")
annotations(x) <- list(name = "keep", what = TRUE)
annotations(x) <- list(name = "category", what = factor("a", levels = letters[1:10]))
edit(x)

## End(Not run)

```

---

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 'subcorpus'  
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,  
  ...  
)  
  
## S4 method for signature 'plpr_subcorpus'  
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 formatting 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(pkg = "RcppCWB", corpus = "REUTERS")

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 'DocumentTermMatrix'
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 separate speeches.

**Usage**

```
as.speeches(.Object, ...)
```

```
## S4 method for signature 'partition'
```

```
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  
)
```

```
## S4 method for signature 'subcorpus'
```

```
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  
)
```

```
## S4 method for signature 'corpus'
```

```
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,  
  subset,  
  mc = FALSE,  
  verbose = TRUE,  
  progress = TRUE  
)
```

```

)

## S4 method for signature 'character'
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

<code>.Object</code>	A partition, or length-one character vector indicating a CWB corpus.
<code>...</code>	Further arguments.
<code>s_attribute_date</code>	A length-one character vector, the s-attribute that provides the dates of sessions.
<code>s_attribute_name</code>	A length-one character vector, the s-attribute that provides the names of speakers.
<code>gap</code>	An integer value, the 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 separate speeches.
<code>mc</code>	Whether to use multicore, defaults to FALSE. If <code>progress</code> is TRUE, argument <code>mc</code> is passed into <code>pblapply</code> as argument <code>cl</code> . If <code>progress</code> is FALSE, <code>mc</code> is passed into <code>mclapply()</code> as argument <code>mc.cores</code> .
<code>verbose</code>	A logical value, defaults to TRUE.
<code>progress</code>	A logical value, whether to show progress bar.
<code>subset</code>	A logical expression evaluated in a temporary <code>data.table</code> with columns 'speaker' and 'date' to define a subset of the entire corpus to be turned into speeches. Usually faster than applying <code>as.speeches()</code> on a partition or subcorpus.

### 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",
  s_attribute_date = "date"
)
summary(speeches)
sp <- corpus("GERMAPARLMINI") %>%
  as.speeches(s_attribute_name = "speaker", s_attribute_date = "date")

sp <- corpus("GERMAPARLMINI") %>%
  as.speeches(
    s_attribute_name = "speaker",
    s_attribute_date = "date",
    subset = {date == as.Date("2009-11-11")},
    progress = FALSE
  )

sp <- corpus("GERMAPARLMINI") %>%
  as.speeches(
    s_attribute_name = "speaker",
    s_attribute_date = "date",
    subset = {date == "2009-11-10" & grepl("Merkel", speaker)},
    progress = FALSE
  )

```

---

as.TermDocumentMatrix *Generate TermDocumentMatrix / DocumentTermMatrix.*

---

## Description

Methods to generate the classes `TermDocumentMatrix` or `DocumentTermMatrix` as defined in the `tm` 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 'corpus'
as.DocumentTermMatrix(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 = NULL, p_attribute = NULL, verbose = TRUE, ...)

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

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

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

## S4 method for signature 'subcorpus_bundle'
as.DocumentTermMatrix(x, p_attribute = 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).
...	Definitions of s-attribute used for subsetting the corpus, compare partition-method.
p_attribute	A p-attribute counting is be based on.
s_attribute	An s-attribute that defines content of columns, or rows.
verbose	A logical value, 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

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.

If `x` is a `partition_bundle`, and argument `col` is not NULL, as `TermDocumentMatrix` is generated based on the column indicated by `col` of the `data.table` with counts in the stat slots of the objects in the bundle. If `col` is NULL, the `p_attribute` indicated by `p_attribute` is decoded, and a count is performed to obtain the values of the resulting `TermDocumentMatrix`. The same procedure applies to get a `DocumentTermMatrix`.

If `x` is a `subcorpus_bundle`, the `p_attribute` provided by argument `p_attribute` is decoded, and a count is performed to obtain the resulting `TermDocumentMatrix` or `DocumentTermMatrix`.

## Value

A `TermDocumentMatrix`, or a `DocumentTermMatrix` object. These classes are defined in the `tm` package, and inherit from the `simple_triplet_matrix`-class defined in the `slam`-package.

## Author(s)

Andreas Blaette

## Examples

```
use("polmineR")

# enriching partition_bundle explicitly
tdm <- partition("GERMAPARLMINI", date = ".*", regex = TRUE) %>%
  partition_bundle(s_attribute = "date") %>%
  enrich(p_attribute = "word") %>%
  as.TermDocumentMatrix(col = "count")

# leave the counting to the as.TermDocumentMatrix-method
tdm <- partition_bundle("GERMAPARLMINI", s_attribute = "date") %>%
  as.TermDocumentMatrix(p_attribute = "word", verbose = FALSE)

# obtain TermDocumentMatrix directly (fastest option)
tdm <- as.TermDocumentMatrix("GERMAPARLMINI", p_attribute = "word", s_attribute = "date")

use(pkg = "RcppCWB", corpus = "REUTERS")

dtm <- corpus("REUTERS") %>%
  split(s_attribute = "id") %>%
  as.TermDocumentMatrix(p_attribute = "word")
```

---

as.VCorpus	<i>Get VCorpus.</i>
------------	---------------------

---

### Description

Retrieve full text for the subcorpora or partition objects in a subcorpus\_bundle or 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 (probable) scenario, it is possible to use a coerce-method (as(YOUROBJECT, "VCorpus")), see examples.

### Examples

```
pb <- partition("GERMAPARLMINI", date = "2009-11-10") %>%
  partition_bundle(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

use(pkg = "RcppCWB", corpus = "REUTERS")
vc <- corpus("REUTERS") %>% split(s_attribute = "id") %>% as("VCorpus")
```

---

blapply	<i>apply a function over a list or bundle</i>
---------	---

---

### Description

Very similar to lapply, but applicable to bundle-objects, in particular. The purpose of the method is to supply a uniform and 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"))
```

---

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'  
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 method for signature 'bundle,ANY,ANY,ANY'  
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, keep.rownames, 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)

## S3 method for class 'bundle'
as.list(x, ...)

## S4 method for signature 'bundle'
get_corpus(x)
```

### Arguments

x	a bundle object
value	character string with a name to be assigned
e1	object 1
e2	object 2
i	integer or character values for indexing 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
keep.rownames	Required argument to safeguard consistency with S3 method definition in the <code>data.table</code> package. Unused in this context.
col	columns of the <code>data.table</code> to use to generate an object.

### Slots

`corpus` The CWB corpus the xobjects 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**

```

party_bundle <- partition_bundle("GERMAPARLMINI", s_attribute = "party")

length(party_bundle)
names(party_bundle)
get_corpus(party_bundle)

party_bundle <- enrich(party_bundle, p_attribute = "word")
summary(party_bundle)

# Indexing bundle objects
reu <- corpus("REUTERS") %>% split(s_attribute = "id")
reu[1:3]
reu[-1]
reu[-(1:10)]
reu["127"]
reu[c("127", "273")]

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(pkg = "RcppCWB", corpus = "REUTERS")

pb <- partition_bundle("REUTERS", s_attribute = "id")
coocs <- cooccurrences(pb, query = "oil", cq = FALSE)
dt <- as.data.table(coocs, col = "11")
m <- as.matrix(dt[, 2:ncol(dt)], rownames = dt[["token"]])

```

---

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 observations, 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

Other statistical methods: [ll\(\)](#), [pmi\(\)](#), [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)

```

**Description**

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

**Usage**

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

```
## S4 method for signature 'slice'
```

```
context(  
  .Object,  
  query,  
  cqp = is.cqp,  
  check = TRUE,  
  left = getOption("polmineR.left"),  
  right = getOption("polmineR.right"),  
  p_attribute = getOption("polmineR.p_attribute"),  
  region = NULL,  
  boundary = NULL,  
  stoplist = NULL,  
  positivelist = NULL,  
  regex = FALSE,  
  count = TRUE,  
  mc = getOption("polmineR.mc"),  
  verbose = FALSE,  
  progress = TRUE,  
  ...  
)
```

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

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

## S4 method for signature 'matrix'
context(
  .Object,
  corpus,
  left,
  right,
  p_attribute,
  region = NULL,
  boundary = NULL
)

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



```

    verbose = FALSE,
    progress = TRUE,
    ...
)

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

## S4 method for signature 'partition_bundle'
context(
  .Object,
  query,
  p_attribute,
  stoplist = NULL,
  positivelist = NULL,
  regex = FALSE,
  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 be 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.

<code>left</code>	A single integer value defining the number of tokens to the left of the query match to include in the context. Advanced usage: (a) If <code>left</code> is a length-one character vector stating an s-attribute, the context will be expanded to the (left) boundary of the region where the match occurs. (b) If <code>left</code> is a named length-one integer vector, this value is the number regions of the structural attribute referred to by the vector's name to the left of the query match that are included in the context.
<code>right</code>	A single integer value, a length-one character vector or a named length-one integer value, with equivalent effects to argument <code>left</code> .
<code>p_attribute</code>	The p-attribute of the query.
<code>region</code>	An s-attribute, given by a length-one character vector. The context of query matches will be expanded to the left and right boundary of the region where the match is located. If arguments <code>left</code> and <code>right</code> are $> 1$ , the left and right boundaries of the respective number of regions will be identified.
<code>boundary</code>	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.
<code>stoplist</code>	Exclude match for query if stopword(s) is/are are present in context. See <code>positivelist</code> for further explanation.
<code>positivelist</code>	A character vector or numeric/integer vector: include a query hit only if token in <code>positivelist</code> is present. If <code>positivelist</code> is a character vector, it may include regular expressions (see parameter <code>regex</code> ).
<code>regex</code>	A logical value, defaults to FALSE - whether <code>stoplist</code> and/or <code>positivelist</code> are regular expressions.
<code>count</code>	logical
<code>mc</code>	Whether to use multicore; if NULL (default), the function will get the value from the options.
<code>verbose</code>	Report progress? A logical value, defaults to TRUE.
<code>progress</code>	A logical value, whether to show progress bar.
<code>corpus</code>	A length-one character vector stating a corpus ID.
<code>complete</code>	enhance completely

### Details

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

If `.Object` is a `matrix`, the `context`-method will call `RcppCWB::region_matrix_context()`, the worker behind the `context()`-method.

### 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. Note that the number of objects in the `context_bundle` may differ from the number of objects in the input `bundle` object: NULL objects that result if no hit is obtained are dropped.

**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,
  stat = 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,
  fn = NULL,
  verbose = TRUE,
  progress = TRUE,
  ...
)

```

### Arguments

<code>x</code>	A context object.
<code>.Object</code>	A context object.
<code>size</code>	An integer indicating sample size.
<code>s_attribute</code>	The s-attribute(s) to add to <code>data.table</code> in slot <code>cpos</code> .
<code>p_attribute</code>	The p-attribute(s) to add to <code>data.table</code> in slot <code>cpos</code> .
<code>decode</code>	A logical value, whether to convert integer ids to expressive strings.
<code>stat</code>	A logical value, whether to generate / update slot <code>stat</code> from the <code>cpos</code> table.
<code>verbose</code>	A logical, whether to be talkative.
<code>...</code>	To maintain backwards compatibility if argument <code>pAttribute</code> is still used.
<code>node</code>	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.
<code>positivelist</code>	Tokens that are required to be present to keep a match.
<code>regex</code>	A logical value, whether arguments <code>positivlist</code> / <code>stoplist</code> are interpreted as regular expressions.
<code>stoplist</code>	Tokens that are used to exclude a match.
<code>fn</code>	A function that will be applied on context tables splitted by <code>match_id</code> .
<code>progress</code>	A logical value, 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 "match\_id", "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 match\_id, cpos, position, word\_id.  
 method A character-vector, statistical test used.  
 call Object of class character, call that generated the object.

**Examples**

```

# Keep matches for 'oil' only if first position to the left is 'crude'
.fn <- function(x) if (x[position == -1L][["word"]] == "crude") x else NULL
crude_oil <- context("REUTERS", "oil") %>%
  enrich(p_attribute = "word", decode = TRUE) %>%
  trim(fn = .fn)

```

---

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	<i>Get cooccurrence statistics.</i>
---------------	-------------------------------------

---

**Description**

Get cooccurrence statistics.

**Usage**

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

## S4 method for signature 'corpus'
cooccurrences(
  .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,
  keep = NULL,
  cpos = NULL,
  method = "ll",
  mc = getOption("polmineR.mc"),
  verbose = FALSE,
  progress = FALSE,
  ...
)

## S4 method for signature 'character'
cooccurrences(
  .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,
    keep = NULL,
    cpos = NULL,
    method = "ll",
    mc = getOption("polmineR.mc"),
    verbose = FALSE,
    progress = FALSE,
    ...
)

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

```

    ...
)

## S4 method for signature 'subcorpus'
cooccurrences(
  .Object,
  query,
  cqp = is.cqp,
  left = getOption("polmineR.left"),
  right = getOption("polmineR.right"),
  p_attribute = getOption("polmineR.p_attribute"),
  boundary = 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 'partition_bundle'
cooccurrences(
  .Object,
  query,
  verbose = FALSE,
  mc = getOption("polmineR.mc"),
  ...
)

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

## S4 method for signature 'remote_corpus'
cooccurrences(.Object, ...)

## S4 method for signature 'remote_subcorpus'
cooccurrences(.Object, ...)

```

### Arguments

<code>.Object</code>	A partition object, or a character vector with a CWB corpus.
<code>...</code>	Further parameters that will be passed into <code>bigmatrix</code> (applies only of <code>big = TRUE</code> ).



query	A query, either a character vector to match a token, or a CQP query.
cqp	Defaults to <code>is.cqp-function</code> , or provide TRUE/FALSE; relevant only if query is not NULL.
p_attribute	The p-attribute of the tokens/the query.
boundary	If provided, it will be checked that the corpus positions of windows do not extend beyond the left and right boundaries of the region defined by the s-attribute where the match occurs.
left	A single integer value defining the number of tokens to the left of the query match to include in the context. Advanced usage: (a) If <code>left</code> is a length-one character vector stating an s-attribute, the context will be expanded to the (left) boundary of the region where the match occurs. (b) If <code>left</code> is a named length-one integer vector, this value is the number regions of the structural attribute referred to by the vector's name to the left of the query match that are included in the context.
right	A single integer value, a length-one character vector or a named length-one integer value, with equivalent effects to argument <code>left</code> .
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 <code>positivelist</code> is present. If <code>positivelist</code> is a character vector, it is assumed to provide regex expressions (incredibly long if the list is long) (relevant only if query is not NULL)
regex	A logical value, whether <code>stoplist/positivelist</code> 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 "II").
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")
use(pkg = "RcppCWB", corpus = "REUTERS")

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 with subset
cooccurrences("REUTERS", query = "oil") %>%
  subset(!is.na(ll)) %>%
  subset(!word %in% tm::stopwords("en")) %>%
  subset(count_coi >= 5) %>%
  subset(ll >= 10.83) %>%
  format()

# generate datatables htmlwidget with buttons for export (Excel & more)
# (alternatively use openxlsx::write.xlsx())
interactive_table <- cooccurrences("REUTERS", query = "oil") %>%
  format() %>%
  DT::datatable(
    extensions = "Buttons",
    options = list(dom = 'Btip', buttons = c("excel", "pdf", "csv"))
  )
if (interactive()) show(interactive_table)
pb <- partition_bundle("GERMAPARLMINI", s_attribute = "speaker")
ps <- count(pb, query = "Deutschland")[Deutschland >= 25][["partition"]]
pb_min <- pb[ps]
y <- cooccurrences(pb_min, query = "Deutschland")
if (interactive()) y[[1]]
if (interactive()) y[[2]]

y2 <- corpus("GERMAPARLMINI") %>%
  subset(speaker %in% c("Hubertus Heil", "Angela Dorothea Merkel")) %>%
  split(s_attribute = "speaker") %>%
  cooccurrences(query = "Deutschland")

```

---

Cooccurrences, corpus-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 'corpus'
Cooccurrences(
  .Object,
  p_attribute,
  left,
  right,
  stoplist = NULL,
  mc = getOption("polmineR.mc"),
  verbose = FALSE,
  progress = FALSE
)

## 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 'slice'
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
)

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

```

### Arguments

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

  # Essentially 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 = getOption("polmineR.villainChars")
  )

## 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", ...)

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

```

### Arguments

<code>x</code>	A Cooccurrences class object.
<code>edge_attributes</code>	Attributes from <code>stat data.table</code> in <code>x</code> to add to edges.
<code>vertex_attributes</code>	Vertex attributes to add to nodes.
<code>as.undirected</code>	Logical, whether to return directed or undirected graph.
<code>drop</code>	A character vector indicating names of nodes to drop from <code>igraph</code> object that is prepared.
<code>...</code>	Further arguments passed into a further call of <code>subset</code> .
<code>by</code>	A features-class object.
<code>.Object</code>	A Cooccurrences-class object.
<code>left</code>	Number of tokens to the left of the node.
<code>right</code>	Number of tokens to the right of the node.
<code>verbose</code>	Logical.
<code>progress</code>	Logical, whether to show progress bar.
<code>col</code>	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 `Cooccurrences`-object to be subsetted by a `features`-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, performed 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.

The `enrich()`-method will add columns `'a_count'` and `'b_count'` to the `data.table` in the `'stat'` slot of the `Cooccurrences` object. If the count for the subcorpus/partition from which the cooccurrences are derived is not yet present, the count is performed first.

## 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**

```

use(pkg = "RcppCWB", corpus = "REUTERS")

X <- Cooccurrences("REUTERS", p_attribute = "word", left = 2L, right = 2L)
m <- as.simple_triplet_matrix(X)
## Not run:
use(pkg = "RcppCWB", corpus = "REUTERS")

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-class

*Corpus class initialization*


---

**Description**

Corpora indexed using the Corpus Workbench (CWB) offer an efficient data structure for large, linguistically annotated corpora. The corpus-class keeps basic information on a CWB corpus. Corresponding to the name of the class, the corpus-method is the initializer for objects of the corpus class. A CWB corpus can also be hosted remotely on an [OpenCPU](#) server. The remote\_corpus class (which inherits from the corpus class) will handle respective information. A (limited) set of polmineR functions and methods can be executed on the corpus on the remote machine from the local R session by calling them on the remote\_corpus object. Calling the corpus-method without an argument will return a data.frame with basic information on the corpora that are available.

**Usage**

```

## S4 method for signature 'character'
corpus(.Object, registry_dir, server = NULL, restricted)

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

```

**Arguments**

.Object	The upper-case ID of a CWB corpus stated by a length-one character vector.
registry_dir	The registry directory with the registry file describing the corpus (length-one character vector). If missing, the C representations of loaded corpora will be evaluated to get the registry directory with the registry file for the corpus.
server	If NULL (default), the corpus is expected to be present locally. If provided, the name of an OpenCPU server (can be an IP address) that hosts a corpus, or several corpora. The corpus-method will then instantiate a remote_corpus object.
restricted	A logical value, whether access to a remote corpus is restricted (TRUE) or not (FALSE).

## Details

Calling `corpus()` will return a `data.frame` listing the corpora available locally and described in the active registry directory, and some basic information on the corpora.

A corpus object is instantiated by passing a corpus ID as argument `.Object`. Following the conventions of the Corpus Workbench (CWB), Corpus IDs are written in upper case. If `.Object` includes lower case letters, the corpus object is instantiated nevertheless, but a warning is issued to prevent bad practice. If `.Object` is not a known corpus, the error message will include a suggestion if there is a potential candidate that can be identified by `agrep`.

A limited set of methods of the `polmineR` package is exposed to be executed on a remote OpenCPU server. As a matter of convenience, the whereabouts of an OpenCPU server hosting a CWB corpus can be stated in an environment variable "OPENCPU\_SERVER". Environment variables for R sessions can be set easily in the `.Renviron` file. A convenient way to do this is to call `usethis::edit_r_environ()`.

## Slots

`corpus` A length-one character vector, the upper-case ID of a CWB corpus.

`registry_dir` Registry directory with registry file describing the corpus.

`data_dir` The directory where binary files of the indexed corpus reside.

`info_file` If available, the info file indicated in the registry file (typically a file named `.info` or `info.md` in the data directory), or NA if not.

`template` Full path to the template containing formatting instructions when showing full text output (`fs_path` object or NA).

`type` If available, the type of the corpus (e.g. "plpr" for a corpus of plenary protocols), or NA.

`name` Full name of the corpus that may be more expressive than the corpus ID.

`encoding` The encoding of the corpus, given as a length-one character vector (usually 'utf8' or 'latin1').

`size` Number of tokens (size) of the corpus, a length-one integer vector.

`server` The URL (can be IP address) of the OpenCPU server. The slot is available only with the `remote_corpus` class inheriting from the `corpus` class.

`user` If the corpus on the server requires authentication, the username.

`password` If the corpus on the server requires authentication, the password.

## See Also

Methods to extract basic information from a corpus object are covered by the `corpus-methods` documentation object. Use the `s_attributes` method to get information on structural attributes. Analytical methods available for corpus objects are [size](#), [count](#), [dispersion](#), [kwic](#), [cooccurrences](#), [as.TermDocumentMatrix](#).

Other classes to manage corpora: [phrases](#), [ranges-class](#), [regions](#), [subcorpus](#)

**Examples**

```

use(pkg = "RcppCWB", corpus = "REUTERS")

# get corpora present locally
y <- corpus()

# initialize corpus object
r <- corpus("REUTERS")
r <- corpus("reuters") # will work, but will result in a warning

# apply core polmineR methods
a <- size(r)
b <- s_attributes(r)
c <- count(r, query = "oil")
d <- dispersion(r, query = "oil", s_attribute = "id")
e <- kwic(r, query = "oil")
f <- cooccurrences(r, query = "oil")

# used corpus initialization in a pipe
y <- corpus("REUTERS") %>% s_attributes()
y <- corpus("REUTERS") %>% count(query = "oil")

# working with a remote corpus
## Not run:
REUTERS <- corpus("REUTERS", server = Sys.getenv("OPENCPU_SERVER"))
count(REUTERS, query = "oil")
size(REUTERS)
kwic(REUTERS, query = "oil")

GERMAPARL <- corpus("GERMAPARL", server = Sys.getenv("OPENCPU_SERVER"))
s_attributes(GERMAPARL)
size(x = GERMAPARL)
count(GERMAPARL, query = "Integration")
kwic(GERMAPARL, query = "Islam")

p <- partition(GERMAPARL, year = 2000)
s_attributes(p, s_attribute = "year")
size(p)
kwic(p, query = "Islam", meta = "date")

GERMAPARL <- corpus("GERMAPARLMINI", server = Sys.getenv("OPENCPU_SERVER"))
s_attrs <- s_attributes(GERMAPARL, s_attribute = "date")
sc <- subset(GERMAPARL, date == "2009-11-10")

## End(Not run)

```

## Description

A set of generic methods is available to extract basic information from objects of the corpus class.

## Usage

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

## S4 method for signature 'corpus'
get_corpus(x)

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

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

## S4 method for signature 'corpus'
get_info(x)

## S4 method for signature 'corpus'
show_info(x)
```

## Arguments

x	An object of class corpus, or inheriting from it.
object	An object of class corpus, or inheriting from it.
name	A (single) s-attribute.

## Details

A corpus object can have a name, which can be retrieved using the name-method.

Use get\_corpus-method to get the corpus ID from the slot corpus of the corpus object.

The show()-method will show basic information on the corpus object.

Applying the \$-method on a corpus will return the values for the s-attribute stated with argument name.

Use get\_info to get the the content of the info file for the corpus (usually in the data directory of the corpus) and return it as a character vector. Returns NULL if there is not info file.

The show\_info-method will get the content of the info file for a corpus, turn it into an html document, and show the result in the viewer pane of RStudio. If the filename of the info file ends on "md", the document is rendered as markdown.

## Examples

```
# get/show information on corpora
corpus("REUTERS") %>% get_info()
corpus("REUTERS") %>% show_info()
```

```

corpus("GERMAPARLMINI") %>% get_info()
corpus("GERMAPARLMINI") %>% show_info()

use(pkg = "RcppCWB", corpus = "REUTERS")

# show-method
if (interactive()) corpus("REUTERS") %>% show()
if (interactive()) corpus("REUTERS") # show is called implicitly

# get corpus ID
corpus("REUTERS") %>% get_corpus()

# use $ to access s_attributes quickly
use("polmineR")
g <- corpus("GERMAPARLMINI")
g$date
corpus("GERMAPARLMINI")$date #
corpus("GERMAPARLMINI") %>% s_attributes(s_attribute = "date") # equivalent

use("polmineR")
sc <- subset("GERMAPARLMINI", date == "2009-10-27")
sc$date

```

---

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,
  phrases = NULL,
  ...

```

```
)

## S4 method for signature 'subcorpus'
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,
  phrases = NULL,
  ...
)

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

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



```

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

## S4 method for signature 'remote_subcorpus'
count(.Object, ...)

```

### Arguments

<code>.Object</code>	A partition or <code>partition_bundle</code> , or a length-one character vector providing the name of a corpus.
<code>...</code>	Further arguments. If <code>.Object</code> is a <code>remote_corpus</code> object, the three dots ( <code>...</code> ) are used to pass arguments. Hence, it is necessary to state the names of all arguments to be passed explicitly.
<code>query</code>	A character vector (one or multiple terms), CQP syntax can be used.
<code>cqp</code>	Either logical (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not (defaults to <code>is.query</code> auxiliary function).
<code>check</code>	A logical value, whether to check validity of CQP query using <code>check_cqp_query</code> .

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.
phrases	A phrases object. If provided, the denoted regions will be concatenated as phrases.
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 `partition_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.

Multiple queries can be used for argument query. Some care may be necessary when summing up the counts for the individual queries. When the CQP syntax is used, different queries may yield the same match result, so that the sum of all individual query matches may overestimate the true number of unique matches. In the case of overlapping matches, a warning message is issued. Collapsing multiple CQP queries into a single query (separating the individual queries by "|" and wrapping everything in round brackets) solves this problem.

### 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](#). The [hits](#) is the worker behind the dispersion method and offers a similar, yet more low-level functionality as compared to the count method. Using the [hits](#) method may be useful to obtain the data required for flexible cross-tabulations.

**Examples**

```
use("polmineR")
use(pkg = "RcppCWB", corpus = "REUTERS")

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,
  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)

sc <- corpus("GERMAPARLMINI") %>% subset(party == "SPD")
phr <- cpos(sc, query = '"Deutsche.*" "Bundestag.*"', cqp = TRUE) %>%
  as.phrases(corpus = "GERMAPARLMINI", enc = "latin1")
cnt <- count(sc, phrases = phr, p_attribute = "word")

# Multiple queries and overlapping query matches. The first count
# operation will issue a warning that matches overlap, see the second
# example for a solution.
corpus("REUTERS") %>%
  count(query = c('".*oil"', '"turmoil"'), cqp = TRUE)
corpus("REUTERS") %>%
  count(query = '".*oil|turmoil"', cqp = 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'
summary(object)

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

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

**Arguments**

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

**Details**

The summary-method in combination with a weighed count-object can be used to perform a dictionary-based sentiment analysis (see examples).

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, the 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](#).

**Examples**

```
# sample for dictionary-based sentiment analysis
weights <- data.table::data.table(
  word = c("gut", "super", "herrlich", "schlecht", "grob", "mies"),
  weight = c(1,1,1,-1,-1,-1)
)
corp <- corpus("GERMAPARLMINI")
sc <- subset(corp, date == "2009-11-11")
```

```

cnt <- count(sc, p_attribute = "word")
cnt <- weigh(cnt, with = weights)
y <- summary(cnt)

# old, partition-based workflow
p <- partition("GERMAPARLMINI", date = "2009-11-11")
p <- enrich(p, p_attribute = "word")
weights <- data.table::data.table(
  word = c("gut", "super", "herrlich", "schlecht", "grob", "mies"),
  weight = c(1,1,1,-1,-1,-1)
)
p <- weigh(p, with = weights)
summary(p)

```

---

cpos

*Get corpus positions for a query or queries.*


---

### Description

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

### Usage

```

cpos(.Object, ...)

## S4 method for signature 'corpus'
cpos(
  .Object,
  query,
  p_attribute = getOption("polmineR.p_attribute"),
  cqp = is.cqp,
  regex = FALSE,
  check = TRUE,
  verbose = TRUE,
  ...
)

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

```

```

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

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

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

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

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

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

```

### Arguments

<code>.Object</code>	A length-one character vector indicating a CWB corpus, a partition object, or a matrix with corpus positions.
<code>...</code>	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 to look up, regular expression or CQP query). Token ids (i.e. integer values) are also accepted. If query is neither a regular expression nor a CQP query, a sanity check removes accidental leading/trailing whitespace, issuing a respective warning.
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).
regex	Interpret query as a regular expression.
check	A logical value, whether to check validity of CQP query using check_cqp_query.
verbose	A logical value, whether to show messages.

### Details

If the `cpos()`-method is applied on a character or partition 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 there are not matches, NULL is returned.

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.

If `.Object` is a hits object, an integer vector is returned with the individual corpus positions.

. If `.Object` is a matrix, it is assumed to be a region matrix, i.e. a two-column matrix with left and right corpus positions in the first and second row, respectively. For many operations, such as decoding the token stream, it is necessary to inflate the denoted regions into a vector of all corpus positions referred to by the regions defined in the matrix. The `cpos`-method for matrix objects will perform this task robustly.

If `.Object` is NULL, the method will return an empty integer vector. Used internally to handle NULL objects that may be returned from the `cpos`-method if no matches are obtained for a query.

### 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.

### Examples

```
use(pkg = "RcppCWB", corpus = "REUTERS")

# looking up single tokens
cpos("REUTERS", query = "oil")
corpus("REUTERS") %>% cpos(query = "oil")
corpus("REUTERS") %>% subset(grepl("saudi-arabia", places)) %>% cpos(query = "oil")
partition("REUTERS", places = "saudi-arabia", regex = TRUE) %>% cpos(query = "oil")
```

```
# using CQP query syntax
cpos("REUTERS", query = '"Saudi" "Arabia"')
corpus("REUTERS") %>% cpos(query = '"Saudi" "Arabia"')
corpus("REUTERS") %>%
  subset(grepl("saudi-arabia", places)) %>%
  cpos(query = '"Saudi" "Arabia"', cqp = TRUE)
partition("REUTERS", places = "saudi-arabia", regex = TRUE) %>%
  cpos(query = '"Saudi" "Arabia"', cqp = TRUE)
```

---

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, warn = TRUE)

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

## Arguments

query	A character vector with at least one CQP query.
warn	A (length-one) logical value, whether to issue a warning if a query may be buggy.
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.
check	A logical value whether to run <code>check_cqp_query()</code> on queries.

## 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 ([https://cwb.sourceforge.io/files/CQP\\_Tutorial.pdf](https://cwb.sourceforge.io/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 corpus or subcorpus.*


---

**Description**

Decode corpus or subcorpus and return class specified by argument to.

**Usage**

```
decode(.Object, ...)

## S4 method for signature 'corpus'
decode(
  .Object,
  to = c("data.table", "Annotation"),
  p_attributes = NULL,
  s_attributes = NULL,
  decode = TRUE,
  verbose = TRUE
)

## S4 method for signature 'character'
decode(
  .Object,
```

```
    to = c("data.table", "Annotation"),
    s_attributes = NULL,
    p_attributes = NULL,
    decode = TRUE,
    verbose = TRUE
)

## S4 method for signature 'slice'
decode(
  .Object,
  to = "data.table",
  s_attributes = NULL,
  p_attributes = NULL,
  decode = TRUE,
  verbose = TRUE
)

## S4 method for signature 'partition'
decode(
  .Object,
  to = "data.table",
  s_attributes = NULL,
  p_attributes = NULL,
  decode = TRUE,
  verbose = TRUE
)

## S4 method for signature 'subcorpus'
decode(
  .Object,
  to = "data.table",
  s_attributes = NULL,
  p_attributes = NULL,
  decode = TRUE,
  verbose = TRUE
)

## S4 method for signature 'integer'
decode(.Object, corpus, p_attributes, boost = NULL)

## S4 method for signature 'data.table'
decode(.Object, corpus, p_attributes)
```

### Arguments

<code>.Object</code>	The corpus or subcorpus to decode.
<code>...</code>	Further arguments.
<code>to</code>	The class of the returned object, stated as a length-one character vector.

<code>p_attributes</code>	The positional attributes to decode. If NULL (default), all positional attributes will be decoded.
<code>s_attributes</code>	The structural attributes to decode. If NULL (default), all structural attributes will be decoded.
<code>decode</code>	A logical value, whether to decode token ids and struc ids to character strings. If FALSE, the values of columns for p- and s-attributes will be integer vectors. If TRUE (default), the respective columns are character vectors.
<code>verbose</code>	A logical value, whether to output progress messages.
<code>corpus</code>	A CWB indexed corpus, either a length-one character vector, or a corpus object.
<code>boost</code>	A length-one logical value, whether to speed up decoding a long vector of token ids by directly by reading in the lexicon file from the data directory of a corpus. If NULL (default), the internal decision rule is that <code>boost</code> will be TRUE if the corpus is larger than 10 000 000 million tokens and more than 5 percent of the corpus are to be decoded.

## Details

The primary purpose of the method is type conversion. By obtaining the corpus or subcorpus in the format specified by the argument `to`, the data can be processed with tools that do not rely on the Corpus Workbench (CWB). Supported output formats are `data.table` (which can be converted to a `data.frame` or `tibble` easily) or an `Annotation` object as defined in the package `NLP`. Another purpose of decoding the corpus can be to rework it, and to re-import it into the CWB (e.g. using the `cwbtools`-package).

An earlier version of the method included an option to decode a single s-attribute, which is not supported any more. See the `s_attribute_decode` function of the package `RcppCWB`.

If `.Object` is an integer vector, it is assumed to be a vector of integer ids of p-attributes. The `decode`-method will translate token ids to string values as efficiently as possible. The approach taken will depend on the corpus size and the share of the corpus that is to be decoded. To decode a large number of integer ids, it is more efficient to read the lexicon file from the data directory directly and to index the lexicon with the ids rather than relying on `RcppCWB::c1_id2str`. The internal decision rule is to use the lexicon file when the corpus is larger than 10 000 000 million tokens and more than 5 percent of the corpus are to be decoded. The encoding of the character vector that is returned will be the coding of the locale (usually ISO-8859-1 on Windows, and UTF-8 on macOS and Linux machines).

The `decode`-method for `data.table` objects will decode token ids (column `'p-attribute_id'`), adding the corresponding string as a new column. If a column `"cpos"` with corpus positions is present, ids are derived for the corpus positions given first. If the `data.table` neither has a column `"cpos"` nor columns with token ids (i.e. column name ending with `"_id"`), the input `data.table` is returned unchanged. Note that columns are added to the `data.table` in an in-place operation to handle memory parsimoniously.

## Value

The return value will correspond to the class specified by argument `to`.

## See Also

To decode a structural attribute, you can use the `s_attributes`-method, setting argument `unique` as `FALSE` and `s_attribute_decode`. See `as.VCorpus` to decode a `partition_bundle` object, returning a `VCorpus` object.

## Examples

```
use("polmineR")
use(pkg = "RcppCWB", corpus = "REUTERS")

# Decode corpus as data.table
dt <- decode("GERMAPARLMINI", to = "data.table")

# Decode corpus selectively
dt <- decode("GERMAPARLMINI", to = "data.table", p_attributes = "word", s_attributes = "party")

# Decode a subcorpus
dt <- corpus("GERMAPARLMINI") %>%
  subset(speaker == "Angela Dorothea Merkel") %>%
  decode(s_attributes = c("speaker", "party", "date"), to = "data.table")

# Decode subcorpus selectively
corpus("GERMAPARLMINI") %>%
  subset(speaker == "Angela Dorothea Merkel") %>%
  decode(to = "data.table", p_attributes = "word", s_attributes = "party")

# Decode partition
P <- partition("REUTERS", places = "kuwait", regex = TRUE)
dt <- decode(P)

# Previous versions of polmineR offered an option to decode a single
# s-attribute. This is how you could proceed to get a table with metadata.
dt <- decode(P, s_attribute = "id", decode = FALSE)
dt[, "word" := NULL]
dt[, {list(cpos_left = min(.SD[["cpos"]]), cpos_right = max(.SD[["cpos"]]))}, by = "id"]

# Decode subcorpus as Annotation object
## Not run:
if (requireNamespace("NLP")){
  library(NLP)
  p <- corpus("GERMAPARLMINI") %>%
    subset(date == "2009-11-10" & speaker == "Angela Dorothea Merkel")
  s <- as(p, "String")
  a <- as(p, "Annotation")

  # The beauty of having this NLP Annotation object is that you can now use
  # the different annotators of the openNLP package. Here, just a short scenario
  # how you can have a look at the tokenized words and the sentences.

  words <- s[a$type == "word"]
  sentences <- s[a$type == "sentence"] # does not yet work perfectly for plenary protocols
```

```

  doc <- as(p, "AnnotatedPlainTextDocument")
}

## End(Not run)

# decode vector of token ids
y <- decode(0:20, corpus = "GERMAPARLMINI", p_attributes = "word")
dt <- data.table::data.table(cpos = cpos("GERMAPARLMINI", query = "Liebe")[,1])
decode(dt, corpus = "GERMAPARLMINI", p_attributes = c("word", "pos"))
y <- dt[, .N, by = c("word", "pos")]

```

---

dispersion	<i>Dispersion of a query or multiple queries.</i>
------------	---

---

### Description

The method returns a `data.table` with the number of matches of a query or multiple queries (optionally frequencies) in a corpus or subcorpus as partitioned by one or two s-attributes.

### Usage

```

dispersion(.Object, ...)

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

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

```

```
    progress = TRUE,
    verbose = FALSE,
    ...
)

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

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

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

```

)

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

## S4 method for signature 'remote_corpus'
dispersion(.Object, ...)

## S4 method for signature 'remote_subcorpus'
dispersion(.Object, ...)

```

### Arguments

<code>.Object</code>	A corpus, subcorpus or partition object or a corpus provided by a character string.
<code>...</code>	Further parameters.
<code>query</code>	A character vector stating one or multiple queries.
<code>s_attribute</code>	A character vector (length 1 or 2) providing s-attributes.
<code>cqp</code>	If logical, whether the query is a CQP query, 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
<code>p_attribute</code>	Length one character vector, the p-attribute that will be looked up (typically 'word' or 'lemma').
<code>freq</code>	A logical value, whether to calculate normalized frequencies.
<code>fill</code>	A logical value, whether to report zero matches. Defaults to TRUE. But note that if there are few matches and many values of the s-attribute(s), the resulting data structure is sparse and potentially bloated.
<code>mc</code>	A logical value, whether to use multicore.
<code>progress</code>	A logical value, whether to show progress.
<code>verbose</code>	A logical value, whether to be verbose.
<code>source</code>	The source of the evaluation the hits reported in <code>.Object</code> are based on, a corpus, subcorpus or partition object.

### Details

Augmenting the `data.table` with zeros for subcorpora that do not yield query matches (argument `fill = TRUE`) may require adding many new columns. A respective warning issued by the `data.table` package is supplemented an additional explanatory note of the `polmineR` package.

**Value**

A data.table.

A data.table.

**Author(s)**

Andreas Blaette

**See Also**

The worker behind the dispersion-method is the hits-method.

count

**Examples**

```
use("polmineR")
dispersion("GERMAPARLMINI", query = "Integration", s_attribute = "date")

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	<i>Get and set encoding.</i>
----------	------------------------------

---

**Description**

Method for textstat objects and classes inheriting from textstat; if object is a character vector, the encoding of the corpus is returned. If called without arguments, the session character set is returned.

**Usage**

```
encoding(object)

encoding(object) <- value

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

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

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

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

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

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

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

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

```
## S4 replacement method for signature 'call'
encoding(object) <- value

## S4 replacement method for signature 'quosure'
encoding(object) <- value
```

### Arguments

object	A textstat or bundle object (or an object inheriting from these classes), or a length-one character vector specifying a corpus. If missing, the method will return the session character set.
value	Value to be assigned.

### Details

encoding() uses `l10n_info()` and `localeToCharset()` (in this order) to determine the session encoding. If `localeToCharset()` returns NA, "UTF-8" is assumed to be the session encoding.

### Value

A length-one character vector with an encoding.

### Examples

```
# Get session charset.
encoding()

# 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 the encoding of a character vector between the native encoding and the encoding of the corpus.

**Usage**

```
as.utf8(x, from)
```

```
as.nativeEnc(x, from)
```

```
as.corpusEnc(x, from = encoding(), corpusEnc)
```

**Arguments**

x	A character to be converted.
from	A character vector describing the encoding of the input character vector.
corpusEnc	A character vector describing the target encoding, i.e. the encoding of the corpus (usually "latin1", "UTF-8")

**Details**

The encoding of a corpus and the encoding of the terminal (the native encoding) may differ, provoking strange 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 the conversion. 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	<i>Enrich an object.</i>
--------	--------------------------

---

**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 <code>partition</code> , <code>partition_bundle</code> or <code>comp</code> 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 'count_bundle'
```

```
features(  
  x,  
  y,  
  included = FALSE,  
  method = "chisquare",  
  verbose = !progress,  
  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")
# Get features of objects in a count_bundle
ref <- corpus("GERMAPARLMINI") %>% count(p_attribute = "word")
cois <- corpus("GERMAPARLMINI") %>%
  subset(speaker %in% c("Angela Dorothea Merkel", "Hubertus Heil")) %>%
  split(s_attribute = "speaker") %>%
  count(p_attribute = "word")
y <- features(cois, ref, included = TRUE, method = "chisquare", progress = TRUE)

```

---

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 template for reconstructing full text.*

---

**Description**

Templates are used to format the markdown/html output of partitions.

**Usage**

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

```
## S4 method for signature 'character'
get_template(.Object, warn = FALSE)
```

```
## S4 method for signature 'corpus'
get_template(.Object, warn = FALSE)
```

```
## S4 method for signature 'subcorpus'
get_template(.Object, warn = FALSE)
```

**Arguments**

.Object	A corpus, subcorpus or partition object, or a length-one character vector with a corpus ID.
...	Further arguments to be defined.
warn	A logical value, whether to issue a warning if template is not available. Defaults to FALSE.

---

get_token_stream	<i>Get Token Stream.</i>
------------------	--------------------------

---

**Description**

Auxiliary method to get the fulltext of a corpus, subcorpora etc. Can be used to export corpus data to other tools.

**Usage**

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

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

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

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

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

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



```

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

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

## S4 method for signature 'regions'
get_token_stream(
  .Object,
  p_attribute = "word",
  collapse = NULL,
  cpos = FALSE,
  split = FALSE,
  ...
)

## S4 method for signature 'partition_bundle'
get_token_stream(
  .Object,
  p_attribute = "word",
  phrases = NULL,
  subset = NULL,
  min_length = NULL,
  collapse = NULL,
  cpos = FALSE,
  decode = TRUE,
  beautify = FALSE,
  verbose = TRUE,
  progress = FALSE,
  mc = FALSE,
  ...
)

```

### Arguments

.Object	Input object.
...	Arguments that will be passed into the <code>get_token_stream</code> -method for a numeric vector, the real worker.
corpus	A CWB indexed corpus.
p_attribute	A character vector, the p-attribute(s) to decode.
subset	An expression applied on p-attributes, using non-standard evaluation. Note that symbols used in the expression may not be used internally (e.g. 'stopwords').
boost	A length-one logical value, whether to speed up decoding a long vector of token ids by directly reading in the lexicon file from the data directory of a corpus. If NULL (default), the internal decision rule is that <code>boost</code> will be TRUE if the corpus is larger than 10 000 000 million tokens and more than 5 percent of the corpus are to be decoded.

encoding	If not NULL (default) a length-one character vector stating an encoding that will be assigned to the (decoded) token stream.
collapse	If not NULL (default), a length-one character string passed into paste to collapse character vector into a single string.
beautify	A (length-one) logical value, whether to adjust whitespace before and after interpunctuation.
cpos	A logical value, whether to return corpus positions as names of the tokens.
cutoff	Maximum number of tokens to be reconstructed.
decode	A (length-one) logical value, whether to decode token ids to character strings. Defaults to TRUE, if FALSE, an integer vector with token ids is returned.
split	A logical value, whether to return a character vector (when split is FALSE, default) or a list of character vectors; each of these vectors will then represent the tokens of a region defined by a row in a regions matrix.
left	Left corpus position.
right	Right corpus position.
phrases	A phrases object. Defined phrases will be concatenated.
min_length	If not NULL (default), an integer value with minimum length of documents required to keep them in the list object that is returned.
verbose	A length-one logical value, whether to show messages.
progress	A length-one logical value, whether to show progress bar.
mc	Number of cores to use. If FALSE (default), only one thread will be used.

## Details

CWB indexed corpora have a fixed order of tokens which is called the *token stream*. Every token is assigned to a unique *corpus position*. Subsets of the (entire) token stream defined by a left and a right corpus position are called *regions*. The `get_token_stream`-method will extract the tokens (for regions) from a corpus.

The primary usage of this method is to return the token stream of a (sub-)corpus as defined by a corpus, subcorpus or partition object. The methods defined for a numeric vector or a (two-column) matrix defining regions (i.e. left and right corpus positions in the first and second column) are the actual workers for this operation.

The `get_token_stream` has been introduced so serve as a worker by higher level methods such as `read`, `html`, and `as.markdown`. It may however be useful for decoding a corpus so that it can be exported to other tools.

## Examples

```
use(pkg = "RcppCWB", corpus = "REUTERS")

# Decode first words of GERMAPARLMINI corpus (first sentence)
get_token_stream(0:9, corpus = "GERMAPARLMINI", p_attribute = "word")

# Decode first sentence and collapse tokens into single string
get_token_stream(0:9, corpus = "GERMAPARLMINI", p_attribute = "word", collapse = " ")
```

```

# Decode regions defined by two-column integer matrix
region_matrix <- matrix(c(0L,9L,10L,25L), ncol = 2, byrow = TRUE)
get_token_stream(region_matrix, corpus = "GERMAPARLMINI", p_attribute = "word", encoding = "latin1")

# Use argument 'beautify' to remove surplus whitespace
get_token_stream(
  region_matrix,
  corpus = "GERMAPARLMINI",
  p_attribute = "word",
  encoding = "latin1",
  collapse = " ", beautify = TRUE
)

# Decode entire corpus (corpus object / specified by corpus ID)
fulltext <- get_token_stream("GERMAPARLMINI", p_attribute = "word")
corpus("GERMAPARLMINI") %>%
  get_token_stream(p_attribute = "word") %>%
  head()

# Decode subcorpus
corpus("REUTERS") %>%
  subset(id == "127") %>%
  get_token_stream(p_attribute = "word") %>%
  head()

# Decode partition_bundle
pb_tokstr <- corpus("REUTERS") %>%
  split(s_attribute = "id") %>%
  get_token_stream(p_attribute = "word")

# Get token stream for partition_bundle
pb <- partition_bundle("REUTERS", s_attribute = "id")
ts_list <- get_token_stream(pb)

# Use two p-attributes
sp <- corpus("GERMAPARLMINI") %>%
  as.speeches(s_attribute_name = "speaker", s_attribute_date = "date", progress = FALSE)
p2 <- get_token_stream(sp, p_attribute = c("word", "pos"), verbose = FALSE)

# Apply filter
p_sub <- get_token_stream(
  sp, p_attribute = c("word", "pos"),
  subset = {!grepl("(\\$.|$|ART)", pos)}
)

# Concatenate phrases and apply filter
queries <- c("'freiheitliche" "Grundordnung"', "'Bundesrepublik" "Deutschland"' )
phr <- corpus("GERMAPARLMINI") %>% cpos(query = queries) %>% as.phrases(corpus = "GERMAPARLMINI")

kill <- tm::stopwords("de")

ts_phr <- get_token_stream(

```

```

sp,
p_attribute = c("word", "pos"),
subset = {!word %in% kill & !grepl("(\\$.|$|ART)", pos)},
phrases = phr,
progress = FALSE,
verbose = FALSE
)

```

---

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 'corpus'
get_type(.Object)

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

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

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

```

### Arguments

<code>.Object</code>	A partition, <code>partition_bundle</code> , 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")
use(pkg = "RcppCWB", corpus = "REUTERS")

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)

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(), regex = FALSE, perl = FALSE, ...)

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

## 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 needs to be a valid color name. It is also possible to pass in a `matrix` with ranges (as returned by `cpos()`).

`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")
use(pkg = "RcppCWB", corpus = "REUTERS")

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
regions <- cpos(P, query = '"prod.*"', cqp = TRUE)
H2 <- highlight(H, highlight = list(yellow = regions))

# the method can be used in pipe
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")))

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

---

hits

*Get hits for query*

---

### Description

Get hits for queries, optionally with s-attribute values.

### Usage

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

```
## S4 method for signature 'corpus'
```

```
hits(  
  .Object,  
  query,  
  cqp = FALSE,  
  check = TRUE,  
  s_attribute,  
  p_attribute = "word",  
  size = FALSE,  
  freq = FALSE,  
  fill = FALSE,  
  mc = 1L,  
  verbose = TRUE,  
  progress = FALSE,  
  ...  
)
```

```
## S4 method for signature 'character'
```

```
hits(  
  .Object,  
  query,  
  cqp = FALSE,  
  check = TRUE,  
  s_attribute,  
  p_attribute = "word",  
  size = FALSE,  
  freq = FALSE,  
  mc = FALSE,  
  verbose = TRUE,  
  progress = TRUE,  
  ...  
)
```

```
## S4 method for signature 'subcorpus'
```

```
hits(  
  .Object,
```

```
query,
cqp = FALSE,
check = TRUE,
s_attribute,
p_attribute = "word",
size = FALSE,
freq = FALSE,
fill = FALSE,
mc = FALSE,
progress = FALSE,
verbose = TRUE,
...
)

## S4 method for signature 'partition'
hits(
  .Object,
  query,
  cqp = FALSE,
  check = TRUE,
  s_attribute,
  p_attribute = "word",
  size = FALSE,
  freq = FALSE,
  fill = 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"),
  s_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, ...)
```

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

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

## Arguments

<code>.Object</code>	A length-one character vector with a corpus ID, a <code>partition</code> or <code>partition_bundle</code> object
<code>...</code>	Further arguments (used for backwards compatibility).
<code>query</code>	A character vector (optionally named, see details) with one or more queries.
<code>cqp</code>	Either a logical value (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not.
<code>check</code>	A logical value, whether to check validity of CQP query using <code>check_cqp_query</code> .
<code>s_attribute</code>	A character vector of s-attributes that will be used to breakdown counts for matches for query/queries.
<code>p_attribute</code>	A character vector stating a p-attribute.
<code>size</code>	A logical value, whether to report the size of subcorpus.
<code>freq</code>	A logical value, whether to report relative frequencies.
<code>fill</code>	A logical value, whether to report counts (optionally frequencies) for combinations of s-attributes where no matches occur.
<code>mc</code>	A logical value, whether to use multicore.
<code>verbose</code>	A logical value, whether to output messages.
<code>progress</code>	A logical value, whether to show progress bar.

## Details

If the character vector provided by `query` is named, these names will be reported 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).

## Value

A hits class object.

## See Also

See the documentation of the hits class ([hits-class](#)) for details.

## Examples

```

use(pkg = "RcppCWB", corpus = "REUTERS")

# get hits for corpus object
y <- corpus("REUTERS") %>% hits(query = "oil")
y <- corpus("REUTERS") %>% hits(query = c("oil", "barrel"))
y <- corpus("REUTERS") %>% hits(query = "oil", s_attribute = "places", freq = TRUE)

# specify corpus by corpus ID
y <- hits("REUTERS", query = "oil")
y <- hits("REUTERS", query = "oil", s_attribute = "places", freq = TRUE)

# get hits for partition
p <- partition("REUTERS", places = "saudi-arabia", regex = TRUE)
y <- hits(p, query = "oil", s_attribute = "id")

# get hits for subcorpus
y <- corpus("REUTERS") %>%
  subset(grep("saudi-arabia", places)) %>%
  hits(query = "oil")

```

---

hits\_class

*S4 class to represent hits for queries.*

---

## Description

S4 class to represent hits for queries.

## 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 with the following columns:

**query** The query (optionally using CQP syntax) that evoked a hit.

**count** Number of matches in corpus/subcorpus.

**freq** Relative frequency of matches in corpus/subcorpus (optional, presence depends on usage of argument freq of the hits method).

**size** Total number of tokens in corpus/subcorpus (optional, presence depends on usage of argument size of the hits method).

If argument `s_attribute` has been used in the call of the `hits` method, the `data.table` will include additional columns with the `s`-attributes. The values in the columns will be the values these `s`-attributes assume. Columns `count`, `freq` and `size` will be based on subcorpora defined by (combinations of) `s`-attributes.

`corpus` A length-one "character" vector, ID of the corpus with hits for query or queries.

`query` Object of class "character", query or queries for

`p_attribute` The `p`-attribute that has been queried, a length-one character vector.

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

`name` Length-one character vector, name of the object.

---

html	<i>Generate html from object.</i>
------	-----------------------------------

---

## Description

Prepare html document to see full text.

## Usage

```
html(object, ...)
```

```
## S4 method for signature 'character'
html(object, corpus, height = NULL)
```

```
## 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 'subcorpus'
html(
  object,
  meta = NULL,
  cpos = TRUE,
  verbose = FALSE,
  cutoff = NULL,
  charoffset = FALSE,
```

```

    beautify = FALSE,
    height = NULL,
    ...
)

## S4 method for signature 'partition_bundle'
html(
  object,
  charoffset = FALSE,
  beautify = TRUE,
  height = NULL,
  progress = TRUE,
  ...
)

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

## S4 method for signature 'remote_subcorpus'
html(object, ...)

```

### Arguments

object	The object the fulltext output will be based on.
...	Further parameters that are passed into <code>as.markdown</code> .
corpus	The ID of the corpus, a length-one character vector.
height	A character vector that will be inserted into the html as an optional height of a scroll box.
meta	Metadata to include in output, if NULL (default), the s-attributes defining a partition will be used.
cpos	Length-one logical value, if TRUE (default), all tokens will be wrapped by elements with id attribute indicating corpus positions.
verbose	Length-one logical value, whether to output progress messages.
cutoff	An integer value, maximum number of tokens to decode from token stream, passed into <code>as.markdown</code> .
charoffset	Length-one logical value, if TRUE, character offset positions are added to elements embracing tokens.
beautify	Length-one logical value, if TRUE, whitespace before interpunctuation will be removed.
progress	Length-one logical value, whether to output progress# bar.
i	An integer value: 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.
type	The partition type.

## Details

Substitutions configured by option `polmineR.mds` are applied to prevent presence of characters that would be misinterpreted as markdown formatting instructions.

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 tool that stores annotations with character offset positions.

## Value

Returns an object of class `html` as used in the `htmltools` package. Methods such as `htmltools::html_print` will be available. The encoding of the html document will be UTF-8 on all systems (including Windows).

## Examples

```
use(pkg = "RcppCWB", corpus = "REUTERS")

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
H <- partition("REUTERS", places = "argentina") %>% html()

# use html-method to get full text where concordance occurs
K <- kwic("REUTERS", query = "barrels")
H <- html(K, i = 1, s_attribute = "id")
H <- html(K, i = 2, s_attribute = "id")
for (i in 1L: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

---

*Perform keyword-in-context (KWIC) analysis.*


---

## Description

Get concordances for the matches for a query / perform keyword-in-context (`kwic`) analysis.

**Usage**

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

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

## S4 method for signature 'corpus'
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",
  region = NULL,
  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",
region = NULL,
boundary = NULL,
cpos = TRUE,
stoplist = NULL,
positivelist = NULL,
regex = FALSE,
verbose = TRUE,
...
)

## S4 method for signature 'remote_corpus'
kwic(.Object, ...)

## S4 method for signature 'remote_partition'
kwic(.Object, ...)

## S4 method for signature 'remote_subcorpus'
kwic(.Object, ...)

## S4 method for signature 'partition_bundle'
kwic(.Object, ..., verbose = FALSE)

## S4 method for signature 'subcorpus_bundle'
kwic(.Object, ...)

```

### Arguments

<code>.Object</code>	A (length-one) character vector with the name of a CWB corpus, a partition or context object.
<code>...</code>	Further arguments, used to ensure backwards compatibility. If <code>.Object</code> is a <code>remote_corpus</code> or <code>remote_partition</code> object, the three dots ( <code>...</code> ) are used to pass arguments. Hence, it is necessary to state the names of all arguments to be passed explicitly.
<code>s_attributes</code>	Structural attributes (s-attributes) to include into output table as meta-information.
<code>cpos</code>	Logical, if TRUE, a <code>data.table</code> with the corpus positions ("cpos") of the hits and their surrounding context will be assigned to the slot "cpos" of the kwic-object that is returned. Defaults to TRUE, as the availability of the <code>cpos-data.table</code> will often be a prerequisite for further operations on the kwic object. Omitting the table may however be useful to minimize memory consumption.
<code>verbose</code>	A logical value, whether to print messages.
<code>query</code>	A query, CQP-syntax can be used.
<code>cqp</code>	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 <code>is.query</code> ).



<code>left</code>	A single integer value defining the number of tokens to the left of the query match to include in the context. Advanced usage: (a) If <code>left</code> is a length-one character vector stating an s-attribute, the context will be expanded to the (left) boundary of the region where the match occurs. (b) If <code>left</code> is a named length-one integer vector, this value is the number regions of the structural attribute referred to by the vector's name to the left of the query match that are included in the context.
<code>right</code>	A single integer value, a length-one character vector or a named length-one integer value, with equivalent effects to argument <code>left</code> .
<code>region</code>	An s-attribute, given by a length-one character vector. The context of query matches will be expanded to the left and right boundary of the region where the match is located. If arguments <code>left</code> and <code>right</code> are $> 1$ , the left and right boundaries of the respective number of regions will be identified.
<code>p_attribute</code>	The p-attribute, defaults to 'word'.
<code>boundary</code>	If provided, a length-one character vector stating an s-attribute that will be used to check the boundaries of the text.
<code>stoplist</code>	Terms or ids to prevent a concordance from occurring in results.
<code>positivelist</code>	Terms or ids required for a concordance to occur in results
<code>regex</code>	Logical, whether <code>stoplist/positivelist</code> is interpreted as regular expression.
<code>check</code>	A logical value, whether to check validity of CQP query using <code>check_cqp_query</code> .

## 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 query produces a lot of matches, the `DT::datatable()` function used to produce output in the Viewer pane of RStudio may issue a warning. Usually, this warning is harmless and can be ignored. Use options(`"polmineR.warn.size" = FALSE`) for turning off this warning.

If a `positivelist` is supplied, only those concordances will be kept that have one of the terms from the `positivelist` occur 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.

If a `negativelist` is supplied, concordances are removed if any of the tokens of the `negativelist` occurs in the context of the query match.

Applying the `kwic`-method on a `partition_bundle` or `subcorpus_bundle` will return a single `kwic` object that includes a column `'subcorpus_name'` with the name of the subcorpus (or `partition`) in the input object where the match for a concordance occurs.

## Value

If there are no matches, or if all (initial) matches are dropped due to the application of a `positivelist`, a `NULL` is returned.

## 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 standard generic methods applicable to [kwic-class](#) objects. It is possible to read the whole text where a query match occurs, see the [read](#)-method. To highlight terms in the context of a query match, see the [highlight](#)-method.

## Examples

```
use("polmineR")
use(pkg = "RcppCWB", corpus = "REUTERS")

# basic usage
K <- kwic("GERMAPARLMINI", "Integration")
if (interactive()) show(K)
oil <- corpus("REUTERS") %>% kwic(query = "oil")
if (interactive()) show(oil)
oil <- corpus("REUTERS") %>%
  kwic(query = "oil") %>%
  highlight(yellow = "crude")
if (interactive()) show(oil)

# increase left and right context and display metadata
K <- kwic(
  "GERMAPARLMINI",
  "Integration", left = 20, right = 20,
  s_attributes = c("date", "speaker", "party")
)
if (interactive()) show(K)

# use CQP syntax for matching
K <- kwic(
  "GERMAPARLMINI",
  '"Integration" [] "(Menschen|Migrant.*|Personen)"', cqp = TRUE,
  left = 20, right = 20,
  s_attributes = c("date", "speaker", "party")
)
if (interactive()) show(K)

# check that boundary of region is not transgressed
K <- kwic(
  "GERMAPARLMINI",
  '"Sehr" "geehrte"', cqp = TRUE,
  left = 100, right = 100,
  boundary = "date"
)
```

```

if (interactive()) show(K)

# use positivelist and highlight matches in context
K <- kwic("GERMAPARLMINI", query = "Integration", positivelist = "[Ee]urop.*", regex = TRUE)
K <- highlight(K, yellow = "[Ee]urop.*", regex = TRUE)

# Apply kwic on partition_bundle/subcorpus_bundle
gparl_2009_11_10_speeches <- corpus("GERMAPARLMINI") %>%
  subset(date == "2009-11-10") %>%
  as.speeches(
    s_attribute_name = "speaker", s_attribute_date = "date",
    progress = FALSE, verbose = FALSE
  )
k <- kwic(gparl_2009_11_10_speeches, query = "Integration")

```

---

kwic-class

*S4 kwic 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'
get_corpus(x)

## S4 method for signature 'kwic'
count(.Object, p_attribute = "word")

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

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

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

## S4 method for signature 'kwic'
knit_print(x, 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_bundle'
merge(x)

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

## S4 method for signature 'kwic'
format(
  x,
  node_color = "blue",
  align = TRUE,
  extra_color = "grey",
  lineview = getOption("polmineR.lineview")
)

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

```

### Arguments

x	A kwic class object.
.Object	A kwic class object.
p_attribute	A length-one character vector supplying a p-attribute.
verbose	A logical value, whether to output debugging messages.
...	Used for backwards compatibility.
object	A kwic class object.
options	Chunk options.
fmt	A format string passed into <code>sprintf</code> 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.
s_attributes	Character vector of s-attributes with metainformation.

<code>extra</code>	An integer value, number of extra tokens to the left and to the right of the windows of tokens to the left and right of a query match that are decoded to be displayed in a kwic output to facilitate interpretation.
<code>table</code>	Logical, whether to turn <code>cpos</code> <code>data.table</code> into <code>data.table</code> in slot <code>stat</code> for output.
<code>node_color</code>	If not NULL, the html color of the node. If supplied, the node will be wrapped in respective html tags.
<code>align</code>	A logical value for preparing kwic output. If TRUE, whether the content of the columns 'left', 'node' and 'right' will be wrapped in html div elements that will align the output right, centered and left, respectively.
<code>extra_color</code>	If extra context has been generated using <code>enrich</code> , the html color of the additional output (defaults to 'grey').
<code>lineview</code>	A logical value, whether to concatenate left context, node and right context when preparing kwic output.

## Details

Applying the `count`-method on a kwic object will return a count object with the evaluation of the left and right context of the match.

The `knit_print()` method will be called by knitr to render kwic objects as a `DataTable` `htmlwidget` when rendering a R Markdown document as html. It will usually be necessary to explicitly state "render = knit\_print" in the chunk options. The option `polmineR.pageLength` controls the number of lines displayed in the resulting `htmlwidget`. Note that including `htmlwidgets` in html documents requires that `pandoc` is installed. To avoid an error, a formatted `data.table` is returned by `knit_print()` if `pandoc` is not available.

The `as.character`-method will return a list of character vectors, concatenating the columns "left", "node" and "right" of the `data.table` in the `stat`-slot of the input kwic-class object. Optionally, the node can be formatted using a format string that is passed into `sprintf`.

The `subset`-method will apply `subset` to the table in the slot `stat`, e.g. 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.

The `format`-method will return a `data.table` that can serve as input for rendering a `htmlwidget`, for instance using `DT::datatable` or `rhandsontable::rhandsontable`. It will include html tags, so ensure that the rendering engine does not obfuscate the html.

## 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 "match\_id", "cpos", "position", "word\_id", "word" and "direction".

`stat` A data.table, 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.

`name` A length-one character vector naming the object.

`annotation_cols` A character vector designating the columns of the data.table in the slot table that are annotations.

### See Also

The constructor for generating kwic objects is the `kwic` method.

### Examples

```
use("polmineR")
K <- kwic("GERMAPARLMINI", "Integration")
get_corpus(K)
length(K)
K_min <- K[1]
K_min <- K[1:5]

# using kwic_bundle class
queries <- c("oil", "prices", "barrel")
li <- lapply(queries, function(q) kwic("REUTERS", query = q))
kb <- as.bundle(li)

# use count-method on kwic object
coi <- kwic("REUTERS", query = "oil") %>%
  count(p_attribute = "word")

# features vs cooccurrences-method (identical results)
ref <- count("REUTERS", p_attribute = "word") %>%
  subset(word != "oil")
slot(ref, "size") <- slot(ref, "size") - count("REUTERS", "oil")[["count"]]
y_features <- features(coi, ref, method = "ll", included = TRUE)
y_cooc <- cooccurrences("REUTERS", query = "oil")

# extract node and left and right context as character vectors
oil <- kwic("REUTERS", query = "oil")
as.character(oil, fmt = NULL)
as.character(oil) # node wrapped into <i> tag by default
as.character(oil, fmt = "<b>%s</b>")

# subsetting kwic objects
oil <- corpus("REUTERS") %>%
  kwic(query = "oil") %>%
  subset(grepl("prices", right))
saudi_arabia <- corpus("REUTERS") %>%
  kwic(query = "Arabia") %>%
  subset(grepl("Saudi", left))
```

```

int_spd <- corpus("GERMAPARLMINI") %>%
  kwic(query = "Integration") %>%
  enrich(s_attribute = "party") %>%
  subset(grepl("SPD", party))

# turn kwic object into data.frame with html tags
int <- corpus("GERMAPARLMINI") %>%
  kwic(query = "Integration")

as.data.frame(int) # Without further metadata

enrich(int, s_attributes = c("date", "speaker", "party")) %>%
  as.data.frame()

# merge bundle of kwic objects into one kwic
reuters <- corpus("REUTERS")
queries <- c("Saudi" "Arabia", "oil", "'barrel.*'")
comb <- lapply(queries, function(qu) kwic(reuters, query = qu)) %>%
  as.bundle() %>%
  merge()

# enrich kwic object
i <- corpus("GERMAPARLMINI") %>%
  kwic(query = "Integration") %>%
  enrich(s_attributes = c("date", "speaker", "party"))

```

**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 observations, 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.

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

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:

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

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 occurs 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*. <https://aclanthology.org/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.

### See Also

Other statistical methods: `chisquare()`, `pmi()`, `t_test()`

### 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
## Not run:
R <- Cooccurrences("REUTERS", left = 5L, right = 5L, p_attribute = "word")
ll(R)
decode(R)
summary(R)

## End(Not run)

# 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)

```

---

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,  
  n = 2,  
  p_attribute = "word",  
  char = NULL,  
  progress = FALSE,  
  ...  
)
```

```
## S4 method for signature 'character'
```

```
ngrams(  
  .Object,  
  n = 2,  
  p_attribute = "word",  
  char = NULL,  
  progress = FALSE,  
  ...  
)
```

```
## S4 method for signature 'partition'
```

```
ngrams(  
  .Object,  
  n = 2,  
  p_attribute = "word",  
  char = NULL,  
  progress = FALSE,  
  ...  
)
```

```
## S4 method for signature 'subcorpus'
```

```
ngrams(  
  .Object,  
  n = 2,
```

```

    p_attribute = "word",
    char = NULL,
    progress = FALSE,
    ...
)

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

## S4 method for signature 'data.table'
ngrams(.Object, n = 2L, p_attribute = "word")

## S4 method for signature 'corpus'
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

<code>.Object</code>	object of class <code>partition</code>
<code>...</code>	Further arguments.
<code>n</code>	number of tokens/characters
<code>p_attribute</code>	the p-attribute to use (can be > 1)
<code>char</code>	If NULL, tokens will be counted, else characters, keeping only those provided by a character vector

progress	logical
mc	A logical value, whether to use multicore, passed into call to blapply (see respective documentation)

### Examples

```

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

# a more complex scenario: get most frequent ADJA/NN-combinations
ngrm <- ngrams(P, n = 2, p_attribute = c("word", "pos"), char = NULL)
ngrm2 <- subset(
  ngrm,
  ngrm[["1_pos"]] == "ADJA" & ngrm[["2_pos"]] == "NN"
)
ngrm2@stat[, "1_pos" := NULL][, "2_pos" := NULL]
ngrm3 <- sort(ngrm2, by = "count")
head(ngrm3)
use(pkg = "RcppCWB", corpus = "REUTERS")
dt <- decode("REUTERS", p_attribute = "word", s_attribute = character(), to = "data.table")
y <- ngrams(dt, n = 3L, p_attribute = "word")

```

---

ngrams_class	<i>Ngrams class.</i>
--------------	----------------------

---

### Description

Ngrams class.

---

noise	<i>detect noise</i>
-------	---------------------

---

### Description

detect noise

### Usage

```

noise(.Object, ...)

## S4 method for signature 'DocumentTermMatrix'
noise(
  .Object,
  minTotal = 2,
  minTfIdfMean = 0.005,
  sparse = 0.995,

```

```

stopwordsLanguage = "german",
minNchar = 2L,
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

<code>.Object</code>	An object of class <code>DocumentTermMatrix</code> .
<code>...</code>	further parameters
<code>minTotal</code>	minimum colsum (for <code>DocumentTermMatrix</code> ) to qualify a term as non-noise
<code>minTfIdfMean</code>	minimum mean value for tf-idf to qualify a term as non-noise
<code>sparse</code>	Will be passed into <code>tm::removeSparseTerms()</code> .
<code>stopwordsLanguage</code>	e.g. "german", to get stopwords defined in the <code>tm</code> package.
<code>minNchar</code>	Minimum number of characters to qualify a term as non-noise.
<code>specialChars</code>	special characters to drop
<code>numbers</code>	regex, to drop numbers
<code>verbose</code>	logical
<code>p_attribute</code>	relevant if applied to a <code>textstat</code> object

### Value

a list

---

ocpu_exec	<i>Execute code on OpenCPU server</i>
-----------	---------------------------------------

---

### Description

ocpu\_exec will execute a function/method `fn` on an OpenCPU server (specified by argument `server`), using three dots (`...`) to pass arguments. It is the worker of methods defined for `remote_corpus`, `remote_subcorpus` and `remote_partition` objects.

### Usage

```
ocpu_exec(fn, corpus, server, restricted = FALSE, do.call = FALSE, ...)
```

### Arguments

<code>fn</code>	Name of the function/method to execute on remote server (length-one character vector).
<code>corpus</code>	A length-one character vector, the id of the corpus to be queried.
<code>server</code>	The IP/URL of the remote OpenCPU server.
<code>restricted</code>	A logical value, whether credentials are required to access the data.
<code>do.call</code>	Logical, if TRUE, the function <code>fn</code> is passed into a call of <code>do.call</code> , which offers some flexibility.
<code>...</code>	Arguments passed into the method/function call.

### Examples

```
## Not run:
# Get polmineR version installed on remote server
ocpu_exec(
  fn = "packageVersion",
  server = Sys.getenv("OPENCPU_SERVER"),
  do.call = TRUE,
  pkg = "polmineR"
)

## End(Not run)
```

---

partition	<i>Initialize a partition.</i>
-----------	--------------------------------

---

### 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 'corpus'
```

```
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 '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 'context'
partition(.Object, node = TRUE)

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

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

```

### Arguments

<code>.Object</code>	A length-one character-vector, the CWB corpus to be used.
<code>...</code>	Arguments to define partition (see examples). If <code>.Object</code> is a <code>remote_corpus</code> or <code>remote_partition</code> object, the three dots ( <code>...</code> ) are used to pass arguments. Hence, it is necessary to state the names of all arguments to be passed explicitly.
<code>def</code>	A named list of character vectors of s-attribute values, the names are the s-attributes (see details and examples)
<code>name</code>	A name for the new partition object, defaults to "".
<code>encoding</code>	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.
<code>p_attribute</code>	The p-attribute(s) for which a count is performed.
<code>regex</code>	A logical value (defaults to FALSE).
<code>xml</code>	Either 'flat' (default) or 'nested'.
<code>decode</code>	Logical, whether to turn token ids to strings (set FALSE to minimize object size / memory consumption) in <code>data.table</code> with counts.
<code>type</code>	A length-one character vector specifying the type of corpus / partition (e.g. "plpr")
<code>mc</code>	Whether to use multicore (for counting terms).
<code>verbose</code>	Logical, whether to be verbose.
<code>slots</code>	Object slots that will be reported columns of <code>data.frame</code> 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 backslashes etc.). If `regex` is `FALSE`, the length of the character vectors can be  $> 1$ , matching s-attributes are identifies with the operator `'%in%'`.

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 by `.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.

### 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	<i>Generate bundle of partitions.</i>
------------------	---------------------------------------

---

## 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 'corpus'
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 '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

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

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

## 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
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.

**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

**Examples**

```
# merge partition_bundle into one partition
gparl <- corpus("GERMAPARLMINI") %>%
  split(s_attribute = "date") %>%
  merge()
use(pkg = "RcppCWB", corpus = "REUTERS")

pb <- partition_bundle("REUTERS", s_attribute = "id")
barplot(pb, las = 2)

sc <- corpus("GERMAPARLMINI") %>%
  subset(date == "2009-11-10") %>%
  split(s_attribute = "speaker") %>%
  barplot(las = 2)
```

---

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'
p_attributes(.Object, p_attribute = NULL, decode = TRUE)

## S4 method for signature 'subcorpus'
p_attributes(.Object, p_attribute = NULL, decode = TRUE)

is.partition(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)

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

```

**Arguments**

<code>.Object</code>	A partition object.
<code>p_attribute</code>	a p-attribute (for enriching) / performing count.
<code>decode</code>	logical value, whether to decode token ids into strings when performing count
<code>x</code>	A partition object.
<code>verbose</code>	logical value, whether to output messages
<code>mc</code>	logical or, if numeric, providing the number of cores
<code>...</code>	further parameters passed into count when calling enrich, and ...
<code>gap</code>	An integer value specifying the minimum gap between regions for performing the split.

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

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

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.

### 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 `integer`, 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

```
p <- partition("GERMAPARLMINI", date = "2009-11-11", speaker = "Norbert Lammert")
name(p) <- "Norbert Lammert"
pb <- split(p, gap = 500L)
summary(pb)
```

---

partition\_to\_string    *Decode as String.*

---

### Description

Decode as String.

### Examples

```
use("polmineR")
p <- partition("GERMAPARLMINI", date = "2009-11-10", speaker = "Angela Dorothea Merkel")
s <- as(p, "String")
```

---

phrases                    *Manage and use phrases*

---

### Description

Class, methods and functionality for processing phrases (lexical units, lexical items, multi-word expressions) beyond the token level. The envisaged workflow at this stage is to detect phrases using the ngrams-method and to generate a phrases class object from the ngrams object using the as.phrases method. This object can be passed into a call of count, see examples. Further methods and functions documented here are used internally, but may be useful.

### Usage

```
## S4 method for signature 'ngrams'
as.phrases(.Object)

## S4 method for signature 'matrix'
as.phrases(.Object, corpus, enc = encoding(corpus))

## S4 method for signature 'phrases'
as.character(x, p_attribute)

concatenate_phrases(dt, phrases, col)
```

### Arguments

.Object	Input object, either a ngrams or a matrix object.
corpus	A length-one character vector, the corpus ID of the corpus from which regions / the data.table representing a decoded corpus is derived.
enc	Encoding of the corpus.
x	A phrases class object.

<code>p_attribute</code>	The positional attribute (p-attribute) to decode.
<code>dt</code>	A <code>data.table</code> .
<code>phrases</code>	A phrases class object.
<code>col</code>	If <code>.Object</code> is a <code>data.table</code> , the column to concatenate.

### Details

The `phrases` considers a phrase as sequence as tokens that can be defined by region, i.e. a left and a right corpus position. This information is kept in a region matrix in the slot "cpos" of the `phrases` class. The `phrases` class inherits from the `regions` class (which inherits from the `corpus` class), without adding further slots.

If `.Object` is an object of class `ngrams`, the `as.phrases`-method will interpret the `ngrams` as CQP queries, look up the matching corpus positions and return an `phrases` object.

If `.Object` is a matrix, the `as.phrases`-method will initialize a `phrases` object. The corpus and the encoding of the corpus will be assigned to the object.

Applying the `as.character`-method on a `phrases` object will return the decoded regions, concatenated using an underscore as separator.

The `concatenate_phrases` function takes a `data.table` (argument `dt`) as input and concatenates phrases in successive rows into a phrase.

### See Also

Other classes to manage corpora: [corpus-class](#), [ranges-class](#), [regions](#), [subcorpus](#)

### Examples

```
# Workflow to create document-term-matrix with phrases

obs <- corpus("GERMAPARLMINI") %>%
  count(p_attribute = "word")

phrases <- corpus("GERMAPARLMINI") %>%
  ngrams(n = 2L, p_attribute = "word") %>%
  pmi(observed = obs) %>%
  subset(ngram_count > 5L) %>%
  subset(1:100) %>%
  as.phrases()

dtm <- corpus("GERMAPARLMINI") %>%
  as.speeches(s_attribute_name = "speaker", s_attribute_date = "date", progress = TRUE) %>%
  count(phrases = phrases, p_attribute = "word", progress = TRUE, verbose = TRUE) %>%
  as.DocumentTermMatrix(col = "count", verbose = FALSE)

grep("erneuerbaren_Energien", colnames(dtm))
grep("verpasste_Chancen", colnames(dtm))

use(pkg = "RcppCWB", corpus = "REUTERS")

# Derive phrases object from an ngrams object
```

```

reuters_phrases <- ngrams("REUTERS", p_attribute = "word", n = 2L) %>%
  pmi(observed = count("REUTERS", p_attribute = "word")) %>%
  subset(ngram_count >= 5L) %>%
  subset(1:25) %>%
  as.phrases()

phr <- as.character(reuters_phrases, p_attribute = "word")

# Derive phrases from explicitly stated CQP queries

cqp_phrase_queries <- c(
  "'oil" "revenue";',
  "'Sheikh" "Aziz";',
  "'Abdul" "Aziz";',
  "'Saudi" "Arabia";',
  "'oil" "markets";'
)

reuters_phrases <- cpos("REUTERS", cqp_phrase_queries, p_attribute = "word") %>%
  as.phrases(corpus = "REUTERS", enc = "latin1")

# Use the concatenate_phrases() function on a data.table

lexical_units_cqp <- c(
  "'Deutsche.*" "Bundestag.*";',
  "'sozial.*" "Gerechtigkeit";',
  "'Ausschuss" "f.r" "Arbeit" "und" "Soziales";',
  "'soziale.*" "Marktwirtschaft";',
  "'freiheitliche.*" "Grundordnung";'
)

phr <- cpos("GERMAPARLMINI", query = lexical_units_cqp, cqp = TRUE) %>%
  as.phrases(corpus = "GERMAPARLMINI", enc = "word")

dt <- corpus("GERMAPARLMINI") %>%
  decode(p_attribute = "word", s_attribute = character(), to = "data.table") %>%
  concatenate_phrases(phrases = phr, col = "word")

dt[word == "Deutschen_Bundestag"]
dt[word == "soziale_Marktwirtschaft"]

```

## Description

Calculate Pointwise Mutual Information as an information-theoretic approach to find collocations.

**Usage**

```
pmi(.Object, ...)
```

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

```
pmi(.Object)
```

```
## S4 method for signature 'Cooccurrences'
```

```
pmi(.Object)
```

```
## S4 method for signature 'ngrams'
```

```
pmi(.Object, observed, p_attribute = p_attributes(.Object)[1])
```

**Arguments**

.Object	An object.
...	Arguments methods may require.
observed	A count-object with the numbers of the observed occurrences of the tokens in the input ngrams object.
p_attribute	The positional attribute which shall be considered. Relevant only if ngrams have been calculated for more than one p-attribute.

**Details**

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

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

The formula is based on maximum likelihood estimates: When we know the number of observations for token  $x$ ,  $o_x$ , the number of observations for token  $y$ ,  $o_y$  and the size of the corpus  $N$ , the probabilities for the tokens  $x$  and  $y$ , and for the co-occurrence of  $x$  and  $y$  are as follows:

$$p(x) = \frac{o_x}{N}$$

$$p(y) = \frac{o_y}{N}$$

The term  $p(x,y)$  is the number of observed co-occurrences of  $x$  and  $y$ .

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](https://en.wikipedia.org/wiki/Pointwise_mutual_information)).

**References**

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

**See Also**

Other statistical methods: [chisquare\(\)](#), [ll\(\)](#), [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)))
use("polmineR")
use(pkg = "RcppCWB", corpus = "REUTERS")

dt <- decode(
  "REUTERS",
  p_attribute = "word",
  s_attribute = character(),
  to = "data.table",
  verbose = FALSE
)
n <- ngrams(dt, n = 2L, p_attribute = "word")
obs <- count("REUTERS", p_attribute = "word")
phrases <- pmi(n, observed = obs)

```

---

polmineR-defunct      *Defunct functionality*

---

**Description**

Methods and functions not in use any more or that have been superseded by renamed functions.

**Usage**

```

store(...)

mail(...)

browse(...)

```

**Arguments**

...                    Any arguments that may be passed into the defunct function/method.

---

polmineR-generics      *Generic methods defined in the polmineR package*

---

**Description**

This documentation object gives an overview over the generic methods defined in the polmineR package that have no individual man page but are documented directly with the classes they are defined for.

**Usage**

```
get_info(x)
```

```
show_info(x)
```

**Arguments**

x                    An S4 class object.

---

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)

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

## S4 method for signature 'slice'
p_attributes(.Object, p_attribute = NULL, decode = TRUE)

## S4 method for signature 'partition_bundle'
p_attributes(.Object, p_attribute = NULL, decode = TRUE)

## S4 method for signature 'remote_corpus'
p_attributes(.Object, ...)

## S4 method for signature 'remote_partition'
p_attributes(.Object, ...)
```

**Arguments**

.Object            A length-one character vector, or a partition object.  
 ...                Arguments passed to get\_token\_stream.  
 p\_attribute        A p-attribute to decode, provided by a length-one character vector.  
 decode            A length-one logical value. Whether to return decoded p-attributes or unique token ids.

**Details**

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.

**References**

Stefan Evert & The OCWB Development Team, CQP Query Language Tutorial, [https://cwb.sourceforge.io/files/CQP\\_Tutorial](https://cwb.sourceforge.io/files/CQP_Tutorial)

**Examples**

```
use(pkg = "RcppCWB", corpus = "REUTERS")

p_attributes("REUTERS")
p_attributes("REUTERS", p_attribute = "word")
merkel <- partition("GERMAPARLMINI", speaker = "Merkel", regex = TRUE)
merkel_words <- p_attributes(merkel, "word")
```

---

ranges

*Get ranges for query.*

---

**Description**

Get ranges (pairs of left and right corpus positions) for queries.

**Usage**

```
ranges(.Object, ...)

## S4 method for signature 'corpus'
ranges(
  .Object,
  query,
  cqp = FALSE,
  check = TRUE,
  p_attribute = "word",
  verbose = TRUE,
  mc = 1L,
  progress = FALSE
)

## S4 method for signature 'character'
ranges(
  .Object,
  query,
  cqp = FALSE,
  check = TRUE,
```



```

    p_attribute = "word",
    verbose = TRUE,
    mc = 1L,
    progress = FALSE
  )

## S4 method for signature 'subcorpus'
ranges(
  .Object,
  query,
  cqp = FALSE,
  check = TRUE,
  p_attribute = "word",
  verbose = TRUE,
  mc = 1L,
  progress = FALSE
)

## S4 method for signature 'partition'
ranges(
  .Object,
  query,
  cqp = FALSE,
  check = TRUE,
  p_attribute = "word",
  verbose = TRUE,
  mc = 1L,
  progress = FALSE
)

```

### Arguments

<code>.Object</code>	A length-one character vector indicating a CWB corpus, a partition object, or a matrix with corpus positions.
<code>...</code>	Used for reasons of backwards compatibility to process arguments that have been renamed (e.g. <code>pAttribute</code> ).
<code>query</code>	A character vector providing one or multiple queries (token to look up, regular expression or CQP query). Token ids (i.e. integer values) are also accepted. If query is neither a regular expression nor a CQP query, a sanity check removes accidental leading/trailing whitespace, issuing a respective warning.
<code>cqp</code>	Either logical (TRUE if query is a CQP query), or a function to check whether query is a CQP query or not (defaults to <code>is.cqp</code> auxiliary function).
<code>check</code>	A logical value, whether to check validity of CQP query using <code>check_cqp_query</code> .
<code>p_attribute</code>	The p-attribute to search. Needs to be stated only if query is not a CQP query. Defaults to NULL.
<code>verbose</code>	A logical value, whether to show messages.

mc	If logical value TRUE, the value of <code>getOption("polmineR.cores")</code> is passed into <code>mclapply</code> or <code>pblapply</code> as the specification of the number of cores to use. It is also possible to supply an integer value with the number of cores directly. Defaults to 1 (no multicore). Relevant only if several queries are to be processed.
progress	A logical value, whether to show a progress bar when processing multiple queries.

---

ranges-class	<i>Ranges of query matches.</i>
--------------	---------------------------------

---

### Description

S4 class to manage ranges of corpus positions for query matches. The class inherits from the classes `regions` and `corpus`.

### Usage

```
## S3 method for class 'ranges'
as.data.table(x, ...)
```

### Arguments

x	A ranges class object.
...	Additional arguments (unused).

### Slots

query	A length-one character string, query used for query matches.
-------	--

### See Also

Other classes to manage corpora: [corpus-class](#), [phrases](#), [regions](#), [subcorpus](#)

---

read	<i>Display full text.</i>
------	---------------------------

---

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

**Arguments**

<code>.Object</code>	an object to be read (partition or partition_bundle)
<code>...</code>	further parameters passed into read

<code>meta</code>	a character vector supplying s-attributes for the metainformation to be printed; if not stated explicitly, session settings will be used
<code>highlight</code>	a named list of character vectors (see details)
<code>tooltips</code>	a named list (names are colors, vectors are tooltips)
<code>verbose</code>	logical
<code>cpos</code>	logical, if TRUE, corpus positions will be assigned (invisibly) to a cpos tag of a html element surrounding the tokens
<code>cutoff</code>	maximum number of tokens to display
<code>template</code>	template to format output
<code>col</code>	column of <code>data.table</code> with terms to be highlighted
<code>partition_bundle</code>	a <code>partition_bundle</code> object
<code>def</code>	a named list used to define a partition (names are s-attributes, vectors are values of s-attributes)
<code>i</code>	if <code>.Object</code> is an object of the classes <code>kwic</code> or <code>hits</code> , the <code>ith</code> <code>kwic</code> line or <code>hit</code> to derive a partition to be inspected from
<code>type</code>	the partition type, see documentation for <code>partition-method</code>

### 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 keyword-in-context display, see [kwic](#).

### Examples

```
use("polmineR")
merkel <- partition("GERMAPARLMINI", date = "2009-11-10", speaker = "Merkel", regex = TRUE)
if (interactive()) read(merkel, meta = c("speaker", "date"))
if (interactive()) 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

Class to store and process the regions of a corpus. Regions are defined by start and end corpus positions and correspond to a set of tokens surrounded by start and end XML tags.

### Usage

```
regions(x, s_attribute)

## S4 method for signature 'corpus'
regions(x, s_attribute)

## S4 method for signature 'subcorpus'
regions(x, s_attribute)

as.regions(x, ...)

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

### Arguments

x	object of class regions
s_attribute	An s-attribute denoted by a length-one character vector for which regions shall be derived.
...	Further arguments.
keep.rownames	Required argument to safeguard consistency with S3 method definition in the data.table package. Unused in this context.
values	values to assign to a column that will be added

### Details

The regions class is a minimal representation of regions and does not include information on the "strucs" (region IDs) that are used internally to obtain values of s-attributes or information, which combination of conditions on s-attributes has been used to obtain regions. This is left to the subcorpus corpus class. Whereas the subcorpus class is associated with the assumption, that a

set of regions is a meaningful sub-unit of a corpus, the `regions` class has a focus on the individual sequences of tokens defined by a structural attribute (such as paragraphs, sentences, named entities).

Information on regions is maintained in the `cpos` slot of the `regions` S4 class: A two-column matrix with begin and end corpus positions (first and second column, respectively). All other slots are inherited from the `corpus` class.

The understanding of "regions" is modelled on the usage of terms by CWB developers. As it is put in the [CQP Interface and Query Language Manual](#): "Matching pairs of XML start and end tags are encoded as token regions, identified by the corpus positions of the first token (immediately following the start tag) and the last token (immediately preceding the end tag) of the region." (p. 6)

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

The `as.data.table` method returns the matrix with corpus positions in the slot `cpos` as a `data.table`.

### Slots

`cpos` A two-column matrix with start and end corpus positions (first and second column, respectively).

### See Also

Other classes to manage corpora: [corpus-class](#), [phrases](#), [ranges-class](#), [subcorpus](#)

### Examples

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

# Get regions matrix as data.table, without / with values
sc <- corpus("REUTERS") %>% subset(grep("saudi-arabia", places))
regions_dt <- as.data.table(sc)
regions_dt <- as.data.table(
  sc,
  values = s_attributes(sc, "id", unique = FALSE)
)
```

---

registry\_move

*Get registry and data directories.*

---

### Description

The Corpus Workbench (CWB) uses a registry directory with plain text 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 directories 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_move(corpus, registry, registry_new, home_dir_new)

registry(pkg = NULL)

data_dir(pkg = NULL)
```

**Arguments**

corpus	The ID of the corpus for which the registry file shall be moved.
registry	The old registry directory.
registry_new	The new registry directory.
home_dir_new	The new home directory.
pkg	A character string with the name of a single package; if NULL (default), the temporary registry and data directory is returned.

**Details**

The `registry_move` is an auxiliary function to create a copy of a registry file in the directory specified by the argument `registry_new`.

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.

The Corpus Workbench may have problems to cope with a registry path that includes registry non-ASCII characters. On Windows, a call to `utils::shortPathName` will generate the short MS-DOS path name that circumvents resulting problems.

Usage of the temporary registry directory can be suppress by setting the environment variable `POLMINER_USE_TMP_REGISTRY` as `'false'`. In this case, the `registry` function will return the environment variable `CORPUS_REGISTRY` unchanged. The `data_dir` function will return the `"indexed_corpus"` directory that is assumed to live in the same parent directory as the registry directory.

**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")
```

---

renamed	<i>Renamed Functions</i>
---------	--------------------------

---

### 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 compatibility.

### Usage

```
sAttributes(...)
pAttributes(...)
getTokenStream(...)
getTerms(...)
getEncoding(...)
partitionBundle(...)
as.partitionBundle(...)

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

## S4 method for signature 'bundle'
corpus(.Object)

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

### Arguments

...	argument that are passed to the renamed function
.Object	A kwic object.

---

size	<i>Get Number of Tokens.</i>
------	------------------------------

---

### Description

The method will get the number of tokens in a corpus, partition or subcorpus, split up by an s-attribute if provided.



**Usage**

```

size(x, ...)

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

## 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 'partition_bundle'
size(x)

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

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

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

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

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

```

**Arguments**

x	An object to get size(s) for.
...	Further arguments (used only for backwards compatibility).
s_attribute	A character vector with s-attributes (one or more).
verbose	A logical value, whether to output messages.

**Details**

One or more s-attributes can be provided to get the dispersion of tokens across one or more dimensions. If s\_attribute is a child of the s-attribute defining a subcorpus or partition, the struc values need to be decoded for all corpus positions, which may take some time.

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**

If `.Object` is a corpus (a corpus object or specified by corpus id), an integer vector if argument `s_attribute` is `NULL`, a two-column `data.table` otherwise (first column is the s-attribute, second column: "size"). If `.Object` is a `subcorpus_bundle` or a `partition_bundle`, a `data.table` (with columns "name" and "size").

**See Also**

See [dispersion](#)-method for counts of hits. The [hits](#) method calls the `size`-method to get sizes of subcorpora.

**Examples**

```
use("polmineR")
use(pkg = "RcppCWB", corpus = "REUTERS")

# for corpus object
corpus("REUTERS") %>% size()
corpus("REUTERS") %>% size(s_attribute = "id")
corpus("GERMAPARLMINI") %>% size(s_attribute = c("date", "party"))

# for corpus specified by ID
size("GERMAPARLMINI")
size("GERMAPARLMINI", s_attribute = "date")
size("GERMAPARLMINI", s_attribute = c("date", "party"))

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

# for subcorpus
sc <- corpus("GERMAPARLMINI") %>% subset(date == "2009-11-11")
size(sc, s_attribute = "speaker")
size(sc, s_attribute = "party")
size(sc, s_attribute = c("speaker", "party"))

# for subcorpus_bundle
subcorpora <- corpus("GERMAPARLMINI") %>% split(s_attribute = "date")
size(subcorpora)
```

---

slice

*Virtual class slice.*

---

**Description**

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

---

subcorpus

*The S4 subcorpus class.*

---

**Description**

Class to manage subcorpora derived from a CWB corpus.

**Usage**

```
## S4 method for signature 'subcorpus'  
summary(object)  
  
## S4 replacement method for signature 'subcorpus'  
name(x) <- value  
  
## S4 method for signature 'subcorpus'  
get_corpus(x)  
  
## S4 method for signature 'subcorpus'  
size(x, s_attribute = NULL, ...)
```

**Arguments**

object	A subcorpus object.
x	A subcorpus object.
value	A character vector to assign as name to slot name of a subcorpus class object.
s_attribute	A character vector with s-attributes (one or more).
...	Arguments passed into size-method. Used only to maintain backwards compatibility.

**Methods (by generic)**

- `summary`: Get named list with basic information for subcorpus object.
- `name<-`: Assign name to a subcorpus object.
- `get_corpus`: Get the corpus ID from the subcorpus object.
- `size`: Get the size of a subcorpus object from the respective slot of the 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 column matrix with integer values).
annotations	Object of class <code>list</code> .
size	Total size (number of tokens) of the subcorpus object (a length-one integer vector). The value is accessible by calling the <code>size</code> -method on the subcorpus-object (see examples).
metadata	Object of class <code>data.frame</code> , metadata information.
strucs	Object of class <code>integer</code> , the strucs defining the subcorpus.
xml	Object of class <code>character</code> , whether the xml is "flat" or "nested".
s_attribute_strucs	Object of class <code>character</code> , the base node.
user	If the corpus on the server requires authentication, the username.
password	If the corpus on the server requires authentication, the password.

**See Also**

Most commonly, a subcorpus is derived from a corpus or a subcorpus using the [subset](#) method. See [size](#) for detailed documentation on how to use the `size`-method. The subcorpus class shares many features with the `partition` class, but it is more parsimonious and does not include information on statistical properties of the subcorpus (i.e. a count table). In line with this logic, the subcorpus class inherits from the `corpus` class, whereas the `partition` class inherits from the `textstat` class.

Other classes to manage corpora: [corpus-class](#), [phrases](#), [ranges-class](#), [regions](#)

**Examples**

```

use("polmineR")

# basic example
r <- corpus("REUTERS")
k <- subset(r, grepl("kuwait", places))
name(k) <- "kuwait"
y <- summary(k)
s <- size(k)

# the same with a magrittr pipe
corpus("REUTERS") %>%
  subset(grepl("kuwait", places)) %>%
  summary()

# subsetting a subcorpus in a pipe
stone <- corpus("GERMAPARLMINI") %>%
  subset(date == "2009-11-10") %>%
  subset(speaker == "Frank-Walter Steinmeier")

# perform count for subcorpus
n <- corpus("REUTERS") %>% subset(grep("kuwait", places)) %>% count(p_attribute = "word")
n <- corpus("REUTERS") %>% subset(grep("saudi-arabia", places)) %>% count('Saudi' "Arabia")

# keyword-in-context analysis (kwic)
k <- corpus("REUTERS") %>% subset(grep("kuwait", places)) %>% kwic("oil")

```

---

subcorpus\_bundle-class

*Bundled subcorpora*


---

**Description**

A `subcorpus_bundle` object combines a set of `subcorpus` objects in a list in the slot objects. The class inherits from the `partition_bundle` and the `bundle` class. Typically, a `subcorpus_bundle` is generated by applying the `split`-method on a `corpus` or `subcorpus`.

**Usage**

```

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

## S4 method for signature 'subcorpus_bundle'
merge(x, name = "", verbose = FALSE)

## S4 method for signature 'subcorpus'
merge(x, y, ...)

```

```

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

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

## S4 method for signature 'subcorpus_bundle'
split(
  x,
  s_attribute,
  prefix = "",
  progress = TRUE,
  mc = getOption("polmineR.mc")
)

```

### Arguments

object	An object of class subcorpus_bundle.
x	A corpus, subcorpus, or subcorpus_bundle object.
name	The name of the new subcorpus object.
verbose	Logical, whether to provide progress information.
y	A subcorpus to be merged with x.
...	Further subcorpus objects to be merged with x and y.
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.

progress	Logical, whether to show progress bar.
type	The type of partition to generate.
xml	A logical value.

### Details

Applying the `split`-method to a `subcorpus_bundle`-object will iterate through the subcorpus, and apply `split` on each subcorpus object in the bundle, splitting it up by the `s`-attribute provided by the argument `s_attribute`. The return value is a `subcorpus_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.

### Examples

```
corpus("REUTERS") %>% split(s_attribute = "id") %>% summary()

# Merge multiple subcorpus objects
a <- corpus("GERMAPARLMINI") %>% subset(date == "2009-10-27")
b <- corpus("GERMAPARLMINI") %>% subset(date == "2009-10-28")
c <- corpus("GERMAPARLMINI") %>% subset(date == "2009-11-10")
y <- merge(a, b, c)
s_attributes(y, "date")
sc <- subset("GERMAPARLMINI", date == "2009-11-11")
b <- split(sc, s_attribute = "speaker")

p <- partition("GERMAPARLMINI", date = "2009-11-11")
y <- partition_bundle(p, s_attribute = "speaker")
gparl <- corpus("GERMAPARLMINI")
b <- split(gparl, s_attribute = "date")
# split up objects in partition_bundle by using partition_bundle-method
use("polmineR")
y <- corpus("GERMAPARLMINI") %>%
  split(s_attribute = "date") %>%
  split(s_attribute = "speaker")

summary(y)
```

### Description

The structural attributes of a corpus (`s`-attributes) can be used to generate subcorpora (i.e. a subcorpus class object) by applying the `subset`-method. To obtain a subcorpus, the `subset`-method can be applied on a corpus represented by a corpus object, a length-one character vector (as a shortcut), and on a subcorpus object.

**Usage**

```
## S4 method for signature 'corpus'
subset(x, subset, regex = FALSE, ...)

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

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

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

**Arguments**

x	A corpus or subcorpus object. A corpus may also be specified by a length-one character vector.
subset	A logical expression indicating elements or rows to keep. The expression may be unevaluated (using quote or bquote).
regex	A logical value. If TRUE, values for s-attributes defined using the three dots (...) are interpreted as regular expressions and passed into a grep call for subsetting a table with the regions and values of structural attributes. If FALSE (the default), values for s-attributes must match exactly.
...	An expression that will be used to create a subcorpus from s-attributes.

**Value**

A subcorpus object. If the expression provided by argument subset includes undefined s-attributes, a warning is issued and the return value is NULL.

**See Also**

The methods applicable for the subcorpus object resulting from subsetting a corpus or subcorpus are described in the documentation of the `\link{subcorpus-class}`. Note that the subset-method can also be applied to `textstat-class` objects (and objects inheriting from this class).

**Examples**

```
use("polmineR")

# examples for standard and non-standard evaluation
a <- corpus("GERMAPARLMINI")

# subsetting a corpus object using non-standard evaluation
sc <- subset(a, speaker == "Angela Dorothea Merkel")
sc <- subset(a, speaker == "Angela Dorothea Merkel" & date == "2009-10-28")
sc <- subset(a, grepl("Merkel", speaker))
sc <- subset(a, grepl("Merkel", speaker) & date == "2009-10-28")
```



```

# subsetting corpus specified by character vector
sc <- subset("GERMAPARLMINI", grepl("Merkel", speaker))
sc <- subset("GERMAPARLMINI", speaker == "Angela Dorothea Merkel")
sc <- subset("GERMAPARLMINI", speaker == "Angela Dorothea Merkel" & date == "2009-10-28")
sc <- subset("GERMAPARLMINI", grepl("Merkel", speaker) & date == "2009-10-28")

# subsetting a corpus using the (old) logic of the partition-method
sc <- subset(a, speaker = "Angela Dorothea Merkel")
sc <- subset(a, speaker = "Angela Dorothea Merkel", date = "2009-10-28")
sc <- subset(a, speaker = "Merkel", regex = TRUE)
sc <- subset(a, speaker = c("Merkel", "Kauder"), regex = TRUE)
sc <- subset(a, speaker = "Merkel", date = "2009-10-28", regex = TRUE)

# providing the value for s-attribute as a variable
who <- "Volker Kauder"
sc <- subset(a, quote(speaker == !!who))

# quoting and quosures necessary when programming against subset
# note how variable who needs to be handled
gparl <- corpus("GERMAPARLMINI")
subcorpora <- lapply(
  c("Angela Dorothea Merkel", "Volker Kauder", "Ronald Pofalla"),
  function(who) subset(gparl, speaker == !!who)
)

```

---

s\_attributes

*Get s-attributes.*


---

## Description

Structural annotations (s-attributes) of a corpus capture meta-information for regions of tokens. The `s_attributes`-method offers high-level access to the s-attributes present in a corpus or subcorpus, or the values of s-attributes in a corpus/partition.

## Usage

```

s_attributes(.Object, ...)

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

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

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

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

```

```

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

## S4 method for signature 'call'
s_attributes(.Object, corpus)

## S4 method for signature 'quosure'
s_attributes(.Object, corpus)

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

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

## S4 method for signature 'data.table'
s_attributes(.Object, corpus, s_attribute, registry)

```

### Arguments

.Object	A corpus, subcorpus, partition object, or a call. A corpus can also be specified by a length-one character vector.
...	To maintain backward compatibility, if argument sAttribute (deprecated) is used. If .Object is a remote_corpus or remote_subcorpus object, the three dots (...) are used to pass arguments. Hence, it is necessary to state the names of all arguments to be passed explicitly.
s_attribute	The name of a specific s-attribute.
unique	A logical value, whether to return unique values.
regex	A regular expression passed into grep to filter return value by applying a regex.
corpus	A corpus-object or a length one character vector denoting a corpus.
registry	The registry directory with the registry file defining corpus. If missing, the registry directory that can be derived using RcppCWB::corpus_registry_dir() is used.

### Details

Importing XML into the Corpus Workbench (CWB) turns elements and element attributes into so-called "s-attributes". There are two basic uses of the s\_attributes-method: If the argument s\_attribute 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-one character vector), the values of the s-attributes available in the corpus/partition are returned.

If argument unique is FALSE, the full sequence of the s\_attributes is returned, which is a useful building block for decoding a corpus.

If argument `s_attributes` is a character providing several s-attributes, the method will return a `data.table`. If `unique` is `TRUE`, all unique combinations of the s-attributes will be reported by the `data.table`.

If the corpus is based on a nested XML structure, the order of items on the `s_attribute` vector matters. The method for corpus objects will take the first s-attribute as the benchmark and assume that further s-attributes are XML ancestors of the node.

If `.Object` is a call or a quosure (defined in the `rlang` package), the `s_attributes`-method will return a character vector with the s-attributes occurring in the call. This usage is relevant internally to implement the subset method to generate a subcorpus using non-standard evaluation. Usually it will not be relevant in an interactive session.

### Value

A character vector (s-attributes, or values of s-attributes).

### Examples

```
use("polmineR")

s_attributes("GERMAPARLMINI")
s_attributes("GERMAPARLMINI", "date") # dates of plenary meetings
s_attributes("GERMAPARLMINI", s_attribute = c("date", "party"))
s_attributes(corpus("GERMAPARLMINI"))
p <- partition("GERMAPARLMINI", date = "2009-11-10")
s_attributes(p)
s_attributes(p, "speaker") # get names of speakers

# Get s-attributes occurring in a call
s_attributes(quote(grep("Merkel", speaker)), corpus = "GERMAPARLMINI")
s_attributes(quote(speaker == "Angela Merkel"), corpus = "GERMAPARLMINI")
s_attributes(quote(speaker != "Angela Merkel"), corpus = "GERMAPARLMINI")
s_attributes(
  quote(speaker == "Angela Merkel" & date == "2009-10-28"),
  corpus = "GERMAPARLMINI"
)

# Get s-attributes from quosure
s_attributes(
  rlang::new_quosure(quote(grep("Merkel", speaker))),
  corpus = "GERMAPARLMINI"
)
```

---

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 'subcorpus'
terms(x, p_attribute, regex = NULL)

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

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

**Arguments**

x	A corpus, partition or subcorpus object, or a length-one character with a corpus id.
p_attribute	The p-attribute to for which to retrieve results (length-one character vector).
regex	Regular expression(s) to filter results (character vector).
robust	A logical value, whether to check for potential failures.

**Examples**

```
use(pkg = "RcppCWB", corpus = "REUTERS")

r <- partition("REUTERS", id = "144")
words <- terms(r, "word")
terms(r, p_attribute = "word", regex = ".*il.*")
```

---

textstat-class	<i>S4 textstat superclass.</i>
----------------	--------------------------------

---

**Description**

The textstat S4 class is the superclass for the classes features, context, and partition. Usually, these subclasses, which are designed to serve a specified analytical purpose, will be used. Common standard generic methods such as head, tail, dim, nrow, colnames are defined for the textstat class and are available for subclasses by inheritance. The core 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. The textstat class inherits from the corpus class, keeping information on the corpus available.

**Usage**

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

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

## S4 replacement method for signature 'textstat'
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, subset)

## 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'
get_corpus(x)

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

restore(file)

cp(x)

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

## Arguments

x                    An object (textstat or class inheriting from textstat).

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
...	Argument that will be passed into a call of the format method on the object x.
e1	A textstat object.
e2	Another textstat object.
subset	A logical expression indicating elements or rows to keep.
.Object	A textstat object.
options	Chunk options.
file	An rds file to restore (filename).

## 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 `knit_print` method will be called by `knitr` to render `textstat` objects or objects inheriting from the `textstat` class as a `DataTable` `htmlwidget` when rendering a R Markdown document as `html`. It will usually be necessary to explicitly state `"render = knit_print"` in the chunk options. The option `polmineR.pagelength` controls the number of lines displayed in the resulting `htmlwidget`. Note that including `htmlwidgets` in `html` documents requires that `pandoc` is installed. To avoid an error, a formatted `data.table` is returned by `knit_print` if `pandoc` is not available.

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`.

Using the reference semantics of `data.table` objects (i.e. inplace modification) has great advantages for memory efficiency. But there may be unexpected behavior when reloading an S4

textstat object (including classes inheriting from textstat) with a data.table in the stat slot. Use restore to copy the data.table once to have a restored object that works for inplace operations after saving / reloading it.

It is not possible to add columns to the data.table in the stat slot of a textclass object, when the object has been saved and loaded using save()/load(). This scenario applies for instance, when the objects of an interactive R session are saved, and loaded when starting the next interactive R session. The cp() function will create a copy of the object, including an explicit copy of the data.table in the stat slot. Inplace modifications of the new object are possible. The function can also be used to avoid unwanted side effects of modifying an object.

## 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.

annotation\_cols A character vector, column names of data.table in slot stat that are annotations.

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

## Examples

```
use(pkg = "polmineR", corpus = "GERMAPARLMINI")
use(pkg = "RcppCWB", corpus = "REUTERS")

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

# generics defined in the polmineR package
x <- count("REUTERS", p_attribute = "word")
name(x) <- "count_reuters"
name(x)
get_corpus(x)

# 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", "11")]
```

```

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)
sc <- partition("GERMAPARLMINI", speaker = "Angela Dorothea Merkel")
cnt <- count(sc, p_attribute = c("word", "pos"))
cnt_min <- subset(cnt, pos %in% c("NN", "ADJA"))
cnt_min <- subset(cnt, pos == "NE")
use(pkg = "RcppCWB", corpus = "REUTERS")

# Get statistics in textstat object as data.table
count_dt <- corpus("REUTERS") %>%
  subset(grep("saudi-arabia", places)) %>%
  count(p_attribute = "word") %>%
  as.data.table()

# textstat objects stored as *.rds files should be loaded using restore().
# Before moving to examples, this is a brief technical dip why this is
# recommended: If we load the *.rds file with readRDS(), the data.table in
# the slot 'stat' will have the pointer '0x0', and the data.table cannot be
# augmented without having been copied previously.

k <- kwic("REUTERS", query = "oil")
kwicfile <- tempfile(fileext = ".rds")
saveRDS(k, file = kwicfile)
problemprone <- readRDS(file = kwicfile)
problemprone@stat[, "newcol" := TRUE]
"newcol" %in% colnames(problemprone@stat) # is FALSE!

attr(problemprone@stat, ".internal.selfref")
identical(attr(problemprone@stat, ".internal.selfref"), new("externalptr"))

# Restore stored S4 object with copy of data.table in 'stat' slot
k <- kwic("REUTERS", query = "oil")
kwicfile <- tempfile(fileext = ".rds")
saveRDS(k, file = kwicfile)

k2 <- restore(kwicfile)
enrich(k2, s_attribute = "id")
"id" %in% colnames(k2) # is TRUE
k <- kwic("REUTERS", query = "oil")
rdata_file <- tempfile(fileext = ".RData")
save(k, file = rdata_file)
rm(k)

load(rdata_file)
k <- cp(k) # now it is possible to columns by reference
enrich(k, s_attribute = "id")
"id" %in% colnames(k)

```



---

tooltips	<i>Add tooltips to text output.</i>
----------	-------------------------------------

---

### 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	A logical value, whether character vector values of argument tooltips are interpreted as regular expressions.

### Examples

```
use(pkg = "RcppCWB", corpus = "REUTERS")

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 ...
h <- P %>%
  html() %>%
  highlight(yellow = c("barrels", "oil", "gas")) %>%
  tooltips(list(yellow = "energy"))
```

---

trim	<i>Trim an object.</i>
------	------------------------

---

### 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	<i>Perform t-test.</i>
--------	------------------------

---

### Description

Compute t-scores to find collocations.

### 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

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$

where  $\mu$  is the mean of the distribution,  $\bar{x}$  the sample mean,  $s^2$  the sample variance, and  $N$  the sample size.

Following Manning and Schuetze (1999), to test whether two tokens (a and b) are a collocation, the sample mean  $\mu$  is the number of observed co-occurrences of a and b divided by corpus size  $N$ :

$$\mu = \frac{o_{ab}}{N}$$

For the mean of the distribution  $\bar{x}$ , maximum likelihood estimates are used. Given that we know the number of observations of token a,  $o_a$ , the number of observations of b,  $o_b$  and the size of the corpus  $N$ , the probabilities for the tokens a and b, and for the co-occurrence of a and b are as follows, if independence is assumed:

$$P(a) = \frac{o_a}{N}$$

$$P(b) = \frac{o_b}{N}$$

$$P(ab) = P(a)P(b)$$

See the examples for a sample calculation of the t-test, and Evert (2005: 83) for a critical discussion of the "highly questionable" assumptions when using the t-test for detecting co-occurrences.

## References

Manning, Christopher D.; Schuetze, Hinrich (1999): *Foundations of Statistical Natural Language Processing*. MIT Press: Cambridge, Mass., pp. 163-166.

Church, Kenneth W. et al. (1991): Using Statistics in Lexical Analysis. In: Uri Zernik (ed.), *Lexical Acquisition*. Hillsdale, NJ:Lawrence Erlbaum, pp. 115-164 doi: [10.4324/97813157853878](https://doi.org/10.4324/97813157853878)

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>

## See Also

Other statistical methods: [chisquare\(\)](#), [ll\(\)](#), [pmi\(\)](#)

## Examples

```
use("polmineR")
y <- cooccurrences("REUTERS", query = "oil", left = 1L, right = 0L, method = "t_test")
# The critical value (for a = 0.005) is 2.579, so "crude" is a collocation
# of "oil" according to t-test.

# A sample calculation
count_oil <- count("REUTERS", query = "oil")
count_crude <- count("REUTERS", query = "crude")
count_crude_oil <- count("REUTERS", query = '"crude" "oil"', cqp = TRUE)

p_crude <- count_crude$count / size("REUTERS")
p_oil <- count_oil$count / size("REUTERS")
p_crude_oil <- p_crude * p_oil

x <- count_crude_oil$count / size("REUTERS")

t_value <- (x - p_crude_oil) / sqrt(x / size("REUTERS"))
# should be identical with previous result:
as.data.frame(subset(y, word == "crude"))$t_test
```

---

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, corpus, lib.loc = .libPaths(), tmp = FALSE, verbose = TRUE)
```

**Arguments**

<code>pkg</code>	A package including at least one CWB indexed corpus.
<code>corpus</code>	A corpus (or corpora) to be loaded selectively.
<code>lib.loc</code>	A character vector with path names of R libraries.
<code>tmp</code>	A logical value, whether to use a temporary data directory.
<code>verbose</code>	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 temporary registry directory, see [registry](#).

**Examples**

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

---

<code>view</code>	<i>Inspect object using View().</i>
-------------------	-------------------------------------

---

**Description**

Inspect object using `View()`.

**Usage**

```
view(.Object, ...)
```

**Arguments**

<code>.Object</code>	an object
<code>...</code>	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

<code>.Object</code>	A matrix, or a count-object.
<code>...</code>	further parameters
<code>method</code>	The kind of weight to apply.
<code>with</code>	A <code>data.table</code> used to weigh p-attributes. A column 'weight' with term weights is required, and columns with the p-attributes of <code>.Object</code> for matching.
<code>progress</code>	Logical, whether to show a progress bar.

## Examples

```
## Not run:  
library(data.table)  
if (require("zoo") && require("devtools")){  
  
# Source in function 'get_sentiws' from a GitHub gist  
gist_url <- 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 use 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

- \* **classes to manage corpora**
  - corpus-class, 43
  - phrases, 114
  - ranges-class, 122
  - regions, 125
  - subcorpus, 131
- \* **datasets**
  - trim, 146
- \* **package**
  - polmineR-package, 4
- \* **statistical methods**
  - chisquare, 20
  - ll, 95
  - pmi, 116
  - t\_test, 147
- \* **textstatistics**
  - chisquare, 20
- +, bundle, bundle-method (bundle-class), 17
- +, bundle, textstat-method (bundle-class), 17
- +, partition\_bundle, ANY-method (partition\_bundle-class), 109
- +, partition\_bundle, partition-method (partition\_bundle-class), 109
- +, partition\_bundle, partition\_bundle-method (partition\_bundle-class), 109
- +, partition\_bundle-method (partition\_bundle-class), 109
- +, textstat, textstat-method (textstat-class), 140
- [, bundle, ANY, ANY, ANY-method (bundle-class), 17
- [, context, ANY, ANY, ANY-method (context-class), 27
- [, context-method (context-class), 27
- [, context\_bundle, ANY, ANY, ANY-method (context\_bundle-class), 29
- [, context\_bundle-method (context\_bundle-class), 29
- [, kwic, ANY, ANY, ANY-method (kwic-class), 91
- [, kwic-method (kwic-class), 91
- [, partition, ANY, ANY, ANY-method (partition\_class), 111
- [, partition-method (partition\_class), 111
- [, partition\_bundle, ANY, ANY, ANY-method (partition\_bundle-class), 109
- [, partition\_bundle-method (partition\_bundle-class), 109
- [, textstat, ANY, ANY, ANY-method (textstat-class), 140
- [[, bundle-method (bundle-class), 17
- [[, context-method (context-class), 27
- [[, context\_bundle-method (context\_bundle-class), 29
- [[, partition\_bundle-method (partition\_bundle-class), 109
- [[, textstat-method (textstat-class), 140
- [[<-, bundle-method (bundle-class), 17
- \$, bundle-method (bundle-class), 17
- \$, corpus-method (corpus-methods), 45
- \$<-, bundle-method (bundle-class), 17
- aggregate, slice-method (slice), 130
- annotations, 6
- annotations, kwic-method (annotations), 6
- annotations, textstat-method (annotations), 6
- annotations<- (annotations), 6
- annotations<-, kwic, list-method (annotations), 6
- annotations<-, textstat, list-method (annotations), 6
- as (as.VCorpus), 16
- as.bundle (textstat-class), 140
- as.bundle, list-method (bundle-class), 17



- as.bundle, textstat-method  
(bundle-class), 17
- as.character, kwic-method (kwic-class),  
91
- as.character, phrases-method (phrases),  
114
- as.corpusEnc (encodings), 66
- as.cqp (cqp), 56
- as.data.frame, cooccurrences\_bundle-method  
(cooccurrences-class), 42
- as.data.frame, kwic-method (kwic-class),  
91
- as.data.frame, textstat-method  
(textstat-class), 140
- as.data.table.bundle (bundle-class), 17
- as.data.table.ranges (ranges-class), 122
- as.data.table.regions (regions), 125
- as.data.table.textstat  
(textstat-class), 140
- as.DataTables, context-method  
(context-class), 27
- as.DataTables, textstat-method  
(textstat-class), 140
- as.DocumentTermMatrix  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, bundle-method  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, character-method  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, context-method  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, corpus-method  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, kwic-method  
(kwic-class), 91
- as.DocumentTermMatrix, partition\_bundle-method  
(as.TermDocumentMatrix), 13
- as.DocumentTermMatrix, subcorpus\_bundle-method  
(as.TermDocumentMatrix), 13
- as.list, bundle-method (bundle-class), 17
- as.list.bundle (bundle-class), 17
- as.markdown, 8
- as.markdown, partition-method  
(as.markdown), 8
- as.markdown, plpr\_partition-method  
(as.markdown), 8
- as.markdown, plpr\_subcorpus-method  
(as.markdown), 8
- as.markdown, subcorpus-method  
(as.markdown), 8
- as.matrix, bundle-method (bundle-class),  
17
- as.matrix, context\_bundle-method  
(context), 22
- as.matrix, partition\_bundle-method  
(partition\_bundle-class), 109
- as.nativeEnc (encodings), 66
- as.partition\_bundle (partition\_class),  
111
- as.partition\_bundle, list-method  
(partition\_bundle-class), 109
- as.partition\_bundle, partition-method  
(partition\_class), 111
- as.partitionBundle (renamed), 128
- as.phrases (phrases), 114
- as.phrases, matrix-method (phrases), 114
- as.phrases, ngrams-method (phrases), 114
- as.regions (regions), 125
- as.regions, context-method  
(context-class), 27
- as.regions, partition-method  
(partition\_class), 111
- as.simple\_triplet\_matrix, Cooccurrences-method  
(Cooccurrences-class), 39
- as.sparseMatrix, 10
- as.sparseMatrix, bundle-method  
(as.sparseMatrix), 10
- as.sparseMatrix, Cooccurrences-method  
(Cooccurrences-class), 39
- as.sparseMatrix, DocumentTermMatrix-method  
(as.sparseMatrix), 10
- as.sparseMatrix, simple\_triplet\_matrix-method  
(as.sparseMatrix), 10
- as.sparseMatrix, TermDocumentMatrix-method  
(as.sparseMatrix), 10
- as.speeches, 11
- as.speeches, character-method  
(as.speeches), 11
- as.speeches, corpus-method  
(as.speeches), 11
- as.speeches, partition-method  
(as.speeches), 11
- as.speeches, subcorpus-method  
(as.speeches), 11
- as.TermDocumentMatrix, 13, 44
- as.TermDocumentMatrix, bundle-method

- (as.TermDocumentMatrix), 13
- as.TermDocumentMatrix, character-method (as.TermDocumentMatrix), 13
- as.TermDocumentMatrix, context-method (as.TermDocumentMatrix), 13
- as.TermDocumentMatrix, kwic-method (kwic-class), 91
- as.TermDocumentMatrix, partition\_bundle-method (as.TermDocumentMatrix), 13
- as.TermDocumentMatrix, subcorpus\_bundle-method (as.TermDocumentMatrix), 13
- as.utf8 (encodings), 66
- as.VCorpus, 16, 60
- as\_igraph (Cooccurrences-class), 39
- as\_igraph, Cooccurrences-method (Cooccurrences-class), 39
  
- barplot, partition\_bundle-method (partition\_bundle-class), 109
- blapply, 16
- blapply, bundle-method (blapply), 16
- blapply, list-method (blapply), 16
- blapply, vector-method (blapply), 16
- browse (polmineR-defunct), 118
- bundle, 109
- bundle (bundle-class), 17
- bundle-class, 17
  
- check\_cqp\_query (cqp), 56
- chisquare, 20, 97, 117, 148
- chisquare, context-method (chisquare), 20
- chisquare, cooccurrences-method (chisquare), 20
- chisquare, features-method (chisquare), 20
- colnames, textstat-method (textstat-class), 140
- concatenate\_phrases (phrases), 114
- context, 22
- context, character-method (context), 22
- context, cooccurrences-method (context), 22
- context, corpus-method (context), 22
- context, matrix-method (context), 22
- context, partition-method (context), 22
- context, partition\_bundle-method (context), 22
- context, slice-method (context), 22
- context, subcorpus-method (context), 22
- context-class, 27
- context\_bundle-class, 29
- Cooccurrences, 41
- Cooccurrences (Cooccurrences, corpus-method), 35
- cooccurrences, 30, 36, 44
- Cooccurrences, character-method (Cooccurrences, corpus-method), 35
- cooccurrences, character-method (cooccurrences), 30
- cooccurrences, context-method (cooccurrences), 30
- cooccurrences, Cooccurrences-method (cooccurrences), 30
- Cooccurrences, corpus-method, 35
- cooccurrences, corpus-method (cooccurrences), 30
- Cooccurrences, partition-method (Cooccurrences, corpus-method), 35
- cooccurrences, partition-method (cooccurrences), 30
- cooccurrences, partition\_bundle-method (cooccurrences), 30
- cooccurrences, remote\_corpus-method (cooccurrences), 30
- cooccurrences, remote\_subcorpus-method (cooccurrences), 30
- Cooccurrences, slice-method (Cooccurrences, corpus-method), 35
- cooccurrences, slice-method (cooccurrences), 30
- Cooccurrences, subcorpus-method (Cooccurrences, corpus-method), 35
- cooccurrences, subcorpus-method (cooccurrences), 30
- Cooccurrences-class, 39
- cooccurrences-class, 42
- cooccurrences\_bundle (cooccurrences-class), 42
- cooccurrences\_bundle-class (cooccurrences-class), 42
- cooccurrences\_reshaped-class (cooccurrences-class), 42

- corpus, [115](#)
- corpus (corpus-class), [43](#)
- corpus, bundle-method (renamed), [128](#)
- corpus, character-method (corpus-class), [43](#)
- corpus, kwic-method (renamed), [128](#)
- corpus, missing-method (corpus-class), [43](#)
- corpus, textstat-method (renamed), [128](#)
- corpus-class, [43](#)
- corpus-methods, [45](#)
- count, [44, 47](#)
- count, character-method (count), [47](#)
- count, context-method (context-class), [27](#)
- count, corpus-method (count), [47](#)
- count, kwic-method (kwic-class), [91](#)
- count, partition-method (count), [47](#)
- count, partition\_bundle-method (count), [47](#)
- count, remote\_corpus-method (count), [47](#)
- count, remote\_subcorpus-method (count), [47](#)
- count, subcorpus-method (count), [47](#)
- count, subcorpus\_bundle-method (count), [47](#)
- count, vector-method (count), [47](#)
- count-class (count\_class), [51](#)
- count-method (count), [47](#)
- count\_bundle-class (count\_class), [51](#)
- count\_class, [51](#)
- cp (textstat-class), [140](#)
- cpos, [53](#)
- cpos, character-method (cpos), [53](#)
- cpos, corpus-method (cpos), [53](#)
- cpos, hits-method (cpos), [53](#)
- cpos, matrix-method (cpos), [53](#)
- cpos, NULL-method (cpos), [53](#)
- cpos, partition-method (cpos), [53](#)
- cpos, slice-method (cpos), [53](#)
- cpos, subcorpus-method (cpos), [53](#)
- cqp, [56](#)
  
- data\_dir (registry\_move), [126](#)
- decode, [57](#)
- decode, character-method (decode), [57](#)
- decode, Cooccurrences-method (Cooccurrences-class), [39](#)
- decode, corpus-method (decode), [57](#)
- decode, data.table-method (decode), [57](#)
- decode, integer-method (decode), [57](#)
- decode, partition-method (decode), [57](#)
- decode, slice-method (decode), [57](#)
- decode, subcorpus-method (decode), [57](#)
- dim, textstat-method (textstat-class), [140](#)
- dispersion, [44, 51, 61, 130](#)
- dispersion, character-method (dispersion), [61](#)
- dispersion, corpus-method (dispersion), [61](#)
- dispersion, hits-method (dispersion), [61](#)
- dispersion, partition-method (dispersion), [61](#)
- dispersion, remote\_corpus-method (dispersion), [61](#)
- dispersion, remote\_subcorpus-method (dispersion), [61](#)
- dispersion, slice-method (dispersion), [61](#)
- dispersion, subcorpus-method (dispersion), [61](#)
- dotplot, [64](#)
- dotplot, features-method (dotplot), [64](#)
- dotplot, features\_ngrams-method (dotplot), [64](#)
- dotplot, partition-method (dotplot), [64](#)
- dotplot, textstat-method (dotplot), [64](#)
  
- edit, textstat-method (annotations), [6](#)
- encoding, [65](#)
- encoding, bundle-method (encoding), [65](#)
- encoding, call-method (encoding), [65](#)
- encoding, character-method (encoding), [65](#)
- encoding, corpus-method (encoding), [65](#)
- encoding, missing-method (encoding), [65](#)
- encoding, quosure-method (encoding), [65](#)
- encoding, subcorpus-method (encoding), [65](#)
- encoding, textstat-method (encoding), [65](#)
- encoding<- (encoding), [65](#)
- encoding<-, call-method (encoding), [65](#)
- encoding<-, quosure-method (encoding), [65](#)
- encodings, [66](#)
- enrich, [67](#)
- enrich, context-method (context-class), [27](#)
- enrich, Cooccurrences-method (Cooccurrences-class), [39](#)
- enrich, kwic-method (kwic-class), [91](#)
- enrich, partition-method (partition\_class), [111](#)

- enrich, partition\_bundle-method  
(partition\_bundle-class), 109
- enrich, method (enrich), 67
- export (partition\_class), 111
- export, partition-method  
(partition\_class), 111
- features, 68
- features, Cooccurrences-method  
(features), 68
- features, count-method (features), 68
- features, count\_bundle-method  
(features), 68
- features, ngrams-method (features), 68
- features, partition-method (features), 68
- features, partition\_bundle-method  
(features), 68
- features-class, 70
- features\_bundle-class (features-class),  
70
- features\_cooccurrences-class  
(features-class), 70
- features\_ngrams-class (features-class),  
70
- flatten (partition\_bundle-class), 109
- format, cooccurrences-method  
(cooccurrences-class), 42
- format, features-method  
(features-class), 70
- format, kwic-method (kwic-class), 91
- format, textstat-method  
(textstat-class), 140
- get\_corpus (corpus-class), 43
- get\_corpus, bundle-method  
(bundle-class), 17
- get\_corpus, corpus-method  
(corpus-methods), 45
- get\_corpus, kwic-method (kwic-class), 91
- get\_corpus, subcorpus-method  
(subcorpus), 131
- get\_corpus, textstat-method  
(textstat-class), 140
- get\_info (polmineR-generics), 118
- get\_info, corpus-method  
(corpus-methods), 45
- get\_template, 71
- get\_template, character-method  
(get\_template), 71
- get\_template, corpus-method  
(get\_template), 71
- get\_template, subcorpus-method  
(get\_template), 71
- get\_token\_stream, 72
- get\_token\_stream, character-method  
(get\_token\_stream), 72
- get\_token\_stream, corpus-method  
(get\_token\_stream), 72
- get\_token\_stream, matrix-method  
(get\_token\_stream), 72
- get\_token\_stream, numeric-method  
(get\_token\_stream), 72
- get\_token\_stream, partition-method  
(get\_token\_stream), 72
- get\_token\_stream, partition\_bundle-method  
(get\_token\_stream), 72
- get\_token\_stream, regions-method  
(get\_token\_stream), 72
- get\_token\_stream, slice-method  
(get\_token\_stream), 72
- get\_token\_stream, subcorpus-method  
(get\_token\_stream), 72
- get\_type, 76
- get\_type, character-method (get\_type), 76
- get\_type, corpus-method (get\_type), 76
- get\_type, partition\_bundle-method  
(get\_type), 76
- get\_type, subcorpus-method (get\_type), 76
- get\_type, subcorpus\_bundle-method  
(get\_type), 76
- getEncoding (renamed), 128
- getTerms (renamed), 128
- getTokenStream (renamed), 128
- head, context-method (context-class), 27
- head, textstat-method (textstat-class),  
140
- highlight, 77, 90
- highlight, character-method (highlight),  
77
- highlight, html-method (highlight), 77
- highlight, kwic-method (highlight), 77
- hist, count-method (count\_class), 51
- hits, 51, 79, 130
- hits, character-method (hits), 79
- hits, context-method (hits), 79
- hits, corpus-method (hits), 79
- hits, partition-method (hits), 79

- hits,partition\_bundle-method (hits), 79
- hits,remote\_corpus-method (hits), 79
- hits,remote\_subcorpus-method (hits), 79
- hits,subcorpus-method (hits), 79
- hits-class (hits\_class), 82
- hits\_class, 82
- html, 83
- html,character-method (html), 83
- html,kwic-method (html), 83
- html,partition-method (html), 83
- html,partition\_bundle-method (html), 83
- html,remote\_subcorpus-method (html), 83
- html,subcorpus-method (html), 83
  
- is.cqp (cqp), 56
- is.partition (partition\_class), 111
  
- knit\_print,kwic-method (kwic-class), 91
- knit\_print,textstat-method (textstat-class), 140
- kwic, 44, 85, 94, 124
- kwic,character-method (kwic), 85
- kwic,context-method (kwic), 85
- kwic,Cooccurrences-method (Cooccurrences-class), 39
- kwic,corpus-method (kwic), 85
- kwic,partition-method (kwic), 85
- kwic,partition\_bundle-method (kwic), 85
- kwic,remote\_corpus-method (kwic), 85
- kwic,remote\_partition-method (kwic), 85
- kwic,remote\_subcorpus-method (kwic), 85
- kwic,slice-method (kwic), 85
- kwic,subcorpus-method (kwic), 85
- kwic,subcorpus\_bundle-method (kwic), 85
- kwic-class, 91
- kwic\_bundle-class (features-class), 70
  
- length,bundle-method (bundle-class), 17
- length,context-method (context-class), 27
- length,count-method (count\_class), 51
- length,kwic-method (kwic-class), 91
- ll, 21, 34, 95, 117, 148
- ll,context-method (ll), 95
- ll,Cooccurrences-method (ll), 95
- ll,cooccurrences-method (ll), 95
- ll,features-method (ll), 95
  
- mail (polmineR-defunct), 118
  
- means, 98
- means,DocumentTermMatrix-method (means), 98
- merge,kwic\_bundle-method (kwic-class), 91
- merge,partition\_bundle-method (partition\_bundle-class), 109
- merge,subcorpus-method (subcorpus\_bundle-class), 133
- merge,subcorpus\_bundle-method (subcorpus\_bundle-class), 133
  
- name (textstat-class), 140
- name,character-method (textstat-class), 140
- name,corpus-method (corpus-methods), 45
- name,textstat-method (textstat-class), 140
- name<- (textstat-class), 140
- name<-,bundle-method (bundle-class), 17
- name<-,subcorpus-method (subcorpus), 131
- name<-,textstat-method (textstat-class), 140
- names,bundle-method (bundle-class), 17
- names,partition\_bundle-method (partition\_bundle-class), 109
- names,textstat-method (textstat-class), 140
- names<-,bundle,vector-method (bundle-class), 17
- ncol,textstat-method (textstat-class), 140
- ngrams, 99
- ngrams,character-method (ngrams), 99
- ngrams,corpus-method (ngrams), 99
- ngrams,data.table-method (ngrams), 99
- ngrams,partition-method (ngrams), 99
- ngrams,partition\_bundle-method (ngrams), 99
- ngrams,subcorpus-method (ngrams), 99
- ngrams-class (ngrams\_class), 101
- ngrams\_class, 101
- noise, 101
- noise,character-method (noise), 101
- noise,DocumentTermMatrix-method (noise), 101
- noise,TermDocumentMatrix-method (noise), 101
- noise,textstat-method (noise), 101

- nrow, textstat-method (textstat-class), 140
- ocpu\_exec, 103
- opencpu (ocpu\_exec), 103
- p\_attributes, 119
- p\_attributes, character-method (p\_attributes), 119
- p\_attributes, context-method (context-class), 27
- p\_attributes, corpus-method (p\_attributes), 119
- p\_attributes, partition-method (partition\_class), 111
- p\_attributes, partition\_bundle-method (p\_attributes), 119
- p\_attributes, remote\_corpus-method (p\_attributes), 119
- p\_attributes, remote\_partition-method (p\_attributes), 119
- p\_attributes, slice-method (p\_attributes), 119
- p\_attributes, subcorpus-method (partition\_class), 111
- p\_attributes, textstat-method (textstat-class), 140
- partition, 104, 109
- partition, character-method (partition), 104
- partition, context-method (partition), 104
- partition, corpus-method (partition), 104
- partition, environment-method (partition), 104
- partition, partition-method (partition), 104
- partition, remote\_corpus-method (partition), 104
- partition, remote\_partition-method (partition), 104
- partition-class (partition\_class), 111
- partition\_bundle, 107
- partition\_bundle, character-method (partition\_bundle), 107
- partition\_bundle, context-method (partition\_bundle), 107
- partition\_bundle, corpus-method (partition\_bundle), 107
- partition\_bundle, environment-method (partition\_bundle-class), 109
- partition\_bundle, partition-method (partition\_bundle), 107
- partition\_bundle, partition\_bundle-method (partition\_bundle), 107
- partition\_bundle-class, 109
- partition\_class, 67, 106, 111
- partition\_to\_string, 114
- partitionBundle (renamed), 128
- pAttributes (renamed), 128
- phrases, 44, 114, 122, 126, 132
- phrases-class (phrases), 114
- plpr\_partition-class (partition\_class), 111
- plpr\_subcorpus-class (subcorpus), 131
- pmi, 21, 97, 116, 148
- pmi, context-method (pmi), 116
- pmi, Cooccurrences-method (pmi), 116
- pmi, ngrams-method (pmi), 116
- polmineR (polmineR-package), 4
- polmineR-defunct, 118
- polmineR-generics, 118
- polmineR-package, 4
- press\_partition-class (partition\_class), 111
- press\_subcorpus-class (subcorpus), 131
- punctuation (trim), 146
- ranges, 120
- ranges, character-method (ranges), 120
- ranges, corpus-method (ranges), 120
- ranges, partition-method (ranges), 120
- ranges, subcorpus-method (ranges), 120
- ranges-class, 122
- read, 90, 122
- read, data.table-method (read), 122
- read, hits-method (read), 122
- read, kwic-method (read), 122
- read, partition-method (read), 122
- read, partition\_bundle-method (read), 122
- read, regions-method (read), 122
- read, subcorpus-method (read), 122
- regions, 44, 115, 122, 125, 132
- regions, corpus-method (regions), 125
- regions, subcorpus-method (regions), 125
- regions-class (regions), 125
- registry, 149
- registry (registry\_move), 126

- registry\_move, 126
- remote\_corpus (corpus-class), 43
- remote\_corpus-class (corpus-class), 43
- remote\_partition-class (partition\_class), 111
- remote\_subcorpus-class (subcorpus), 131
- renamed, 128
- restore (textstat-class), 140
- round, textstat-method (textstat-class), 140
- rownames, textstat-method (textstat-class), 140
  
- s\_attribute\_decode, 60
- s\_attributes, 44, 60, 137
- s\_attributes, call-method (s\_attributes), 137
- s\_attributes, character-method (s\_attributes), 137
- s\_attributes, corpus-method (s\_attributes), 137
- s\_attributes, data.table-method (s\_attributes), 137
- s\_attributes, partition-method (s\_attributes), 137
- s\_attributes, partition\_bundle-method (partition\_bundle-class), 109
- s\_attributes, quosure-method (s\_attributes), 137
- s\_attributes, remote\_corpus-method (s\_attributes), 137
- s\_attributes, remote\_partition-method (s\_attributes), 137
- s\_attributes, slice-method (s\_attributes), 137
- s\_attributes, subcorpus-method (s\_attributes), 137
- sample, bundle-method (bundle-class), 17
- sample, context-method (context-class), 27
- sample, hits-method (hits\_class), 82
- sample, kwic-method (kwic-class), 91
- sAttributes (renamed), 128
- show, context-method (context-class), 27
- show, context\_bundle-method (context\_bundle-class), 29
- show, cooccurrences-method (cooccurrences-class), 42
- show, corpus-method (corpus-methods), 45
- show, features-method (features-class), 70
- show, html-method (html), 83
- show, kwic-method (kwic-class), 91
- show, partition-method (partition\_class), 111
- show, partition\_bundle-method (partition\_bundle-class), 109
- show, subcorpus\_bundle-method (subcorpus\_bundle-class), 133
- show, textstat-method (textstat-class), 140
- show\_info (polmineR-generics), 118
- show\_info, corpus-method (corpus-methods), 45
- size, 44, 128, 132
- size, character-method (size), 128
- size, corpus-method (size), 128
- size, DocumentTermMatrix-method (size), 128
- size, features-method (size), 128
- size, partition-method (size), 128
- size, partition\_bundle-method (size), 128
- size, remote\_corpus-method (size), 128
- size, remote\_partition-method (size), 128
- size, slice-method (size), 128
- size, subcorpus-method (subcorpus), 131
- size, TermDocumentMatrix-method (size), 128
- slice, 130
- slice-class (slice), 130
- sort, textstat-method (textstat-class), 140
- split (partition\_class), 111
- split, corpus-method (subcorpus\_bundle-class), 133
- split, partition-method (partition\_class), 111
- split, subcorpus-method (subcorpus\_bundle-class), 133
- split, subcorpus\_bundle-method (subcorpus\_bundle-class), 133
- store (polmineR-defunct), 118
- subcorpus, 44, 115, 122, 126, 131
- subcorpus-class (subcorpus), 131
- subcorpus\_bundle-class, 133
- subset, 132, 135
- subset, bundle-method (bundle-class), 17

- subset, character-method (subset), 135
- subset, Cooccurrences-method (Cooccurrences-class), 39
- subset, corpus-method (subset), 135
- subset, kwic-method (kwic-class), 91
- subset, remote\_corpus-method (subset), 135
- subset, subcorpus-method (subset), 135
- subset, textstat-method (textstat-class), 140
- summary, context-method (context-class), 27
- summary, context\_bundle-method (context\_bundle-class), 29
- summary, count-method (count\_class), 51
- summary, features-method (features-class), 70
- summary, features\_bundle-method (features-class), 70
- summary, partition\_bundle-method (partition\_bundle-class), 109
- summary, subcorpus-method (subcorpus), 131
  
- t\_test, 21, 97, 117, 147
- t\_test, context-method (t\_test), 147
- tail, textstat-method (textstat-class), 140
- terms, 139
- terms, character-method (terms), 139
- terms, corpus-method (terms), 139
- terms, partition-method (terms), 139
- terms, slice-method (terms), 139
- terms, subcorpus-method (terms), 139
- textstat-class, 140
- tooltips, 145
- tooltips, character-method (tooltips), 145
- tooltips, html-method (tooltips), 145
- tooltips, kwic-method (tooltips), 145
- trim, 146
- trim, context-method (context-class), 27
- trim, DocumentTermMatrix-method (trim), 146
- trim, TermDocumentMatrix-method (trim), 146
- trim-method (trim), 146
  
- unique, bundle-method (bundle-class), 17
  
- use, 148
  
- view, 149
- view, cooccurrences-method (cooccurrences-class), 42
- view, cooccurrences\_reshaped-method (cooccurrences-class), 42
- view, features-method (features-class), 70
- view, kwic-method (kwic-class), 91
- view, textstat-method (textstat-class), 140
- viewer, 7
  
- weigh, 150
- weigh, count-method (weigh), 150
- weigh, count\_bundle-method (weigh), 150
- weigh, DocumentTermMatrix-method (weigh), 150
- weigh, TermDocumentMatrix-method (weigh), 150
  
- zoom (corpus-class), 43