Package 'tm.plugin.koRpus'

May 18, 2021

Type Package

Title Full Corpus Support for the 'koRpus' Package

Description Enhances 'koRpus' text object classes and methods to also support large corpora. Hierarchical ordering of corpus texts into arbitrary categories will be preserved. Provided classes and methods also improve the ability of using the 'koRpus' package together with the 'tm' package. To ask for help, report bugs, suggest feature improvements, or discuss the global development of the package, please subscribe to the koRpus-dev mailing list (<https://korpusml.reaktanz.de>).

Author m.eik michalke [aut, cre]

Maintainer m.eik michalke <meik.michalke@hhu.de>

Depends R (>= 3.5.0),koRpus (>= 0.13-1),sylly (>= 0.1-6)

Imports methods,parallel,tm,NLP

Suggests koRpus.lang.en,testthat,knitr,rmarkdown

VignetteBuilder knitr

URL https://reaktanz.de/?c=hacking&s=koRpus

BugReports https://github.com/unDocUMeantIt/tm.plugin.koRpus/issues

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

Version 0.4-2

Date 2021-05-17

RoxygenNote 7.1.1

Collate '01_class_01_kRp.corpus.R'
'02_method_01_kRp.corpus-class_readability.R'
'02_method_02_kRp.corpus-class_hyphen.R'
'02_method_03_kRp.corpus-class_lex.div.R'
'02_method_04_kRp.corpus-class_read.corp.custom.R'
'02_method_05_kRp.corpus-class_freq.analysis.R'
'02_method_06_kRp.corpus-class_summary.R'

'02_method_07_kRp.corpus-class_correct.R'
'02_method_08_kRp.corpus-class_query.R'
'02_method_09_kRp.corpus-class_filterByClass.R'
'02_method_10_kRp.corpus-class_jumbleWords.R'
'02_method_11_kRp.corpus-class_clozeDelete.R'
'02_method_12_kRp.corpus-class_cTest.R'
'02_method_13_kRp.corpus-class_textTransform.R'
'02_method_14_kRp.corpus-class_docTermMatrix.R'
'02_method_15_kRp.corpus-class_split_by_doc_id.R'
'02_method_20_kRp.corpus_get_set_is.R'
'02_method_21_kRp.corpus-class_show.R' 'corpus_files.R'
'deprecated.R' 'kRpSource.R' 'readCorpus.R'
'tm.plugin.koRpus-internal.R' 'tm.plugin.koRpus-package.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-18 12:50:02 UTC

R topics documented:

`tm.plugin.koRpus-package`

Full Corpus Support for the 'koRpus' Package

Description

Enhances 'koRpus' text object classes and methods to also support large corpora. Hierarchical ordering of corpus texts into arbitrary categories will be preserved. Provided classes and methods also improve the ability of using the 'koRpus' package together with the 'tm' package. To ask for help, report bugs, suggest feature improvements, or discuss the global development of the package, please subscribe to the koRpus-dev mailing list (<https://korpusml.reaktanz.de>).

Details

The DESCRIPTION file:

Package:	tm.plugin.koRpus
Type:	Package
Version:	0.4-2
Date:	2021-05-17
Depends:	R (>= 3.5.0),koRpus (>= 0.13-1),sylly (>= 0.1-6)
Encoding:	UTF-8
License:	GPL (>= 3)
LazyLoad:	yes
URL:	https://reaktanz.de/?c=hacking&s=koRpus

Author(s)

m.eik michalke [aut, cre]

Maintainer: m.eik michalke <meik.michalke@hhu.de>

See Also

Useful links:

- https://reaktanz.de/?c=hacking&s=koRpus
- Report bugs at https://github.com/unDocUMeantIt/tm.plugin.koRpus/issues

---

`clozeDelete,kRp.corpus-method`

Apply clozeDelete() to all texts in kRp.corpus objects

Description

This method calls `clozeDelete` on all tagged text objects inside the given `obj` object (using `mclapply`).

Usage

```
## S4 method for signature 'kRp.corpus'
clozeDelete(obj, mc.cores = getOption("mc.cores", 1L), ...)
```

Arguments

obj	An object of class kRp.corpus.
mc.cores	The number of cores to use for parallelization, see mclapply.
...	options to pass through to clozeDelete.

Value

An object of the same class as obj.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  head(taggedText(myCorpus), n=10)
  myCorpus <- clozeDelete(myCorpus)
  head(taggedText(myCorpus), n=10)
} else {}
```

---

corpusTagged *Deprecated functions and methods*

Description

These functions were used in earlier versions of the package but either replaced or removed.

Usage

```
corpusTagged(obj, ...)

corpusTTR(obj, ...)

corpusLevel(...)

corpusCategory(...)

corpusID(...)

corpusPath(...)
```

Arguments

obj	No longer used.
...	No longer used.

---

corpus_files *Get a comprehensive data frame describing the files of your corpus*

Description

The function translates the hierarchy defintion given into a data frame with one row for each file, including the generated document ID.

Usage

```
corpus_files(
  dir,
  hierarchy = list(),
  fsep = .Platform$file.sep,
  full_list = FALSE
)
```

Arguments

dir	File path to the root directory of the text corpus, or a TIF[1] compliant data frame.
hierarchy	A named list of named character vectors describing the directory hierarchy level by level. If TRUE instead, the hierarchy structure is taken directly from the directory tree. See section Hierarchy of readCorpus for details.
fsep	Character string defining the path separator to use.
full_list	Logical, see return value.

Value

Either a data frame with columns `doc_id`, `file`, `path` and one further factor column for each hierarchy level, or (if `full_list=TRUE`) a list containing that data frame (`all_files`) and also data frames describing the hierarchy by given names (`hier_names`), directories (`hier_dirs`) and relative paths (`hier_paths`).

References

[1] Text Interchange Formats (https://github.com/ropensci/tif)

Examples

```
myCorpusFiles <- corpus_files(
  dir=file.path(
    path.package("tm.plugin.koRpus"), "examples", "corpus"
  ),
  hierarchy=list(
    Topic=c(
      Winner="Reality Winner",
      Edwards="Natalie Edwards"
    ),
    Source=c(
      Wikipedia_prev="Wikipedia (old)",
      Wikipedia_new="Wikipedia (new)"
    )
  )
)
```

---

correct.hyph,kRp.corpus-method

Methods to correct kRp.corpus objects

Description

These methods enable you to correct errors that occurred during automatic processing, e.g., wrong hyphenation.

Usage

```
## S4 method for signature 'kRp.corpus'
correct.hyph(obj, word = NULL, hyphen = NULL, cache = TRUE)
```

Arguments

`obj`	An object of class `kRp.corpus`.
`word`	A character string, the (possibly incorrectly hyphenated) `word` entry to be replaced with `hyphen`.

hyphen	A character string, the new manually hyphenated version of word. Mustn't contain anything other than characters of word plus the hyphenation mark "-".
cache	Logical, if TRUE, the given hyphenation will be added to the sessions' hyphenation cache. Existing entries for the same word will be replaced.

Details

For details on what these methods do on a per text object basis, please refer to the documentation of correct.hyph in the sylly package.

Value

An object of the same class as obj.

---

cTest,kRp.corpus-method

Apply cTest() to all texts in kRp.corpus objects

Description

This method calls cTest on all tagged text objects inside the given obj object (using mclapply).

Usage

```
## S4 method for signature 'kRp.corpus'
cTest(obj, mc.cores = getOption("mc.cores", 1L), ...)
```

Arguments

obj	An object of class kRp.corpus.
mc.cores	The number of cores to use for parallelization, see mclapply.
...	options to pass through to cTest.

Value

An object of the same class as obj.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
```

```
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  taggedText(myCorpus)[20:30,]
  myCorpus <- cTest(myCorpus)
  taggedText(myCorpus)[20:30,]
} else {}
```

`docTermMatrix,kRp.corpus-method`

Generate a document-term matrix from a corpus object

Description

Calculates a sparse document-term matrix calculated from a given object of class `kRp.corpus` and adds it to the object's feature list. You can also calculate the term frequency inverted document frequency value (tf-idf) for each term.

Usage

```
## S4 method for signature 'kRp.corpus'
docTermMatrix(
  obj,
  terms = "token",
  case.sens = FALSE,
  tfidf = FALSE,
  as.feature = TRUE
)
```

Arguments

`obj`	An object of class `kRp.corpus`.
`terms`	A character string defining the `tokens` column to be used for calculating the matrix.
`case.sens`	Logical, whether terms should be counted case sensitive.
`tfidf`	Logical, if `TRUE` calculates term frequency–inverse document frequency (tf-idf) values instead of absolute frequency.
`as.feature`	Logical, whether the output should be just the sparse matrix or the input object with that matrix added as a feature. Use `corpusDocTermMatrix` to get the matrix from such an aggregated object.

Details

The settings of `terms`, `case.sens`, and `tfidf` will be stored in the object's `meta` slot, so you can use `corpusMeta(...,"doc_term_matrix")` to fetch it.

See the examples to learn how to limit the analysis to desired word classes.

Value

Either an object of the input class or a sparse matrix of class `dgCMatrix`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  # get the document-term frequencies in a sparse matrix
  myDTMatrix <- docTermMatrix(myCorpus, as.feature=FALSE)

  # combine with filterByClass() to, e.g.,  exclude all punctuation
  myDTMatrix <- docTermMatrix(filterByClass(myCorpus), as.feature=FALSE)

  # instead of absolute frequencies, get the tf-idf values
  myDTMatrix <- docTermMatrix(
    filterByClass(myCorpus),
    tfidf=TRUE,
    as.feature=FALSE
  )
} else {}
```

`filterByClass,kRp.corpus-method`

Apply filterByClass() to all texts in kRp.corpus objects

Description

This method calls filterByClass on all tagged text objects inside the given txt object (using mclapply).

Usage

```
## S4 method for signature 'kRp.corpus'
filterByClass(txt, mc.cores = getOption("mc.cores", 1L), ...)
```

Arguments

txt	An object of class kRp.corpus.
mc.cores	The number of cores to use for parallelization, see mclapply.
...	options to pass through to filterByClass.

Value

An object of the same class as txt.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  head(taggedText(myCorpus), n=10)
  # remove all punctuation
  myCorpus <- filterByClass(myCorpus)
  head(taggedText(myCorpus), n=10)
} else {}
```

`freq.analysis,kRp.corpus-method` *Apply freq.analysis() to all texts in kRp.corpus objects*

Description

This method calls `freq.analysis` on all tagged text objects inside the given `txt.file` object.

Usage

```
## S4 method for signature 'kRp.corpus'
freq.analysis(txt.file, ...)
```

Arguments

`txt.file`	An object of class `kRp.corpus`.
`...`	options to pass through to `freq.analysis`.

Details

If `corp.freq` was not specified but a valid object of class `kRp.corp.freq` is found in the `freq` slot of `txt.file`, it is used automatically. That is the case if you called `read.corp.custom` on the object previously.

Value

An object of the same class as `txt.file`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  myCorpus <- read.corp.custom(myCorpus)
```

```
  myCorpus <- freq.analysis(myCorpus)
  corpusFreq(myCorpus)
} else {}
```

---

hyphen,kRp.corpus-method

Apply hyphen() to all texts in kRp.corpus objects

Description

This method calls hyphen on all tagged text objects inside the given words object (using mclapply).

Usage

```
## S4 method for signature 'kRp.corpus'
hyphen(words, mc.cores = getOption("mc.cores", 1L), quiet = TRUE,
      ...)
```

Arguments

words	An object of class kRp.corpus.
mc.cores	The number of cores to use for parallelization, see mclapply.
quiet	Logical, if FALSE shows a status bar for the hyphenation process of each text.
...	options to pass through to hyphen.

Value

An object of the same class as words.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Winner", "Wikipedia_new"
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  myCorpus <- hyphen(myCorpus)
} else {}
```

jumbleWords,kRp.corpus-method
Apply jumbleWords() to all texts in kRp.corpus objects

Description

This method calls jumbleWords on all tagged text objects inside the given words object (using mclapply).

Usage

```
## S4 method for signature 'kRp.corpus'
jumbleWords(words, mc.cores = getOption("mc.cores", 1L), ...)
```

Arguments

words	An object of class kRp.corpus.
mc.cores	The number of cores to use for parallelization, see mclapply.
...	options to pass through to jumbleWords.

Value

An object of the same class as words.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  head(taggedText(myCorpus), n=10)
  myCorpus <- jumbleWords(myCorpus)
  head(taggedText(myCorpus), n=10)
} else {}
```

kRp.corpus,-class *S4 Class kRp.corpus*

Description

Objects of this class can contain full text corpora in a hierachical structure. It supports both the tm package's Corpus class and koRpus' own object classes and stores them in separated slots.

Details

Objects should be created using the readCorpus function.

Slots

lang A character string, naming the language that is assumed for the tokenized texts in this object.

desc A named list of descriptive statistics of the tagged texts.

meta A named list. Can be used to store meta information. Currently, no particular format is defined.

raw A list of objects of class Corpus.

tokens A data frame as used for the tokens slot in objects of class kRp.text. In addition to the columns usually found in those objects, this data frame also has a factor column for each hierarchical category defined (if any).

features A named logical vector, indicating which features are available in this object's feat_list slot. Common features are listed in the description of the feat_list slot.

feat_list A named list with optional analysis results or other content as used by the defined features:

- hierarchy A named list of named character vectors describing the directory hierarchy level by level.
- hyphen A named list of objects of class kRp.hyphen.
- readability A named list of objects of class kRp.readability.
- lex_div A named list of objects of class kRp.TTR.
- freq The freq.analysis slot of a kRp.txt.freq class object after freq.analysis was called.
- corp_freq An object of class kRp.corp.freq, e.g., results of a call to read.corp.custom.
- diff A named list of diff features of a kRp.text object after a method like textTransform was called.
- summary A summary data frame for the full corpus, including descriptive statistics on all texts, as well as results of analyses like readability and lexical diversity, if available.
- doc_term_matrix A sparse document-term matrix, as produced by docTermMatrix.
- stopwords A numeric vector with the total number of stopwords in each text, if stopwords were analyzed during tokenizing or POS tagging.

See the getter and setter methods for easy access to these sub-slots. There can actually be any number of additional features, the above is just a list of those already defined by this package.

Contructor function

Should you need to manually generate objects of this class (which should rarely be the case), the contructor function `kRp.corpus(...)` can be used instead of `new("kRp.corpus",...)`. Whenever possible, stick to `readCorpus`.

Note

There is also `getter and setter methods` for objects of this class.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )
} else {}

# manual creation
emptyCorpus <- kRp.corpus()
```

---

kRpSource *A source function for tm*

Description

An rather untested attempt to sketch a `Source` function for `tm`. Supposed to be used to translate tagged `koRpus` objects into `tm` objects.

Usage

```
kRpSource(obj, encoding = "UTF-8")
```

Arguments

obj	An object of class kRp.text (a class union for tagged text objects).
encoding	Character string, defining the character encoding of the object.

Details

Also provided are the methods getElem and pGetElem for S3 class kRpSource.

Value

An object of class Source, also inheriting class kRpSource.

---

lex.div,kRp.corpus-method

Apply lex.div() to all texts in kRp.corpus objects

Description

This method calls lex.div on all tagged text objects inside the given txt object (using mclapply).

Usage

```
## S4 method for signature 'kRp.corpus'
lex.div(
  txt,
  summary = TRUE,
  mc.cores = getOption("mc.cores", 1L),
  char = "",
  quiet = TRUE,
  ...
)
```

Arguments

txt	An object of class kRp.corpus.
summary	Logical, determines if the summary slot should automatically be updated by calling summary on the result.
mc.cores	The number of cores to use for parallelization, see mclapply.
char	Character vector to specify measures of which characteristics should be computed, see lex.div for details.
quiet	Logical, if FALSE shows a status bar for some measures of each text, see lex.div for details.
...	options to pass through to lex.div.

Value

An object of the same class as `txt`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )
  myCorpus <- lex.div(myCorpus)
  corpusSummary(myCorpus)
} else {}
```

---

query,kRp.corpus-method

Apply query() to all texts in kRp.corpus objects

Description

This method calls `query` on all tagged text objects inside the given object.

Usage

```
## S4 method for signature 'kRp.corpus'
query(
  obj,
  var,
  query,
  rel = "eq",
  as.df = TRUE,
  ignore.case = TRUE,
  perl = FALSE,
  regexp_var = "token"
)
```

Arguments

obj	An object of class kRp.corpus.
var	A character string naming a column in the tagged text. If set to "regexp", grepl is called on the column specified by regexp_var.
query	A character vector (for words), regular expression, or single number naming values to be matched in the variable. Can also be a vector of two numbers to query a range of frequency data, or a list of named lists for multiple queries (see "Query lists" section of query).
rel	A character string defining the relation of the queried value and desired results. Must either be "eq" (equal, the default), "gt" (greater than), "ge" (greater of equal), "lt" (less than) or "le" (less or equal). If var="word", is always interpreted as "eq"
as.df	Logical, if TRUE, returns a data frame, otherwise an object of the input class.
ignore.case	Logical, passed through to grepl if var="regexp".
perl	Logical, passed through to grepl if var="regexp".
regexp_var	A character string naming the column to query if var="regexp".

Value

Depending on the arguments, might include whole objects, lists, single values etc.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  query(myCorpus, var="lttr", query="7", rel="gt")
} else {}
```

`read.corp.custom,kRp.corpus-method`
Apply read.corp.custom() to all texts in kRp.corpus objects

Description

This method calls `read.corp.custom` on all tagged text objects inside the given `corpus` object.

Usage

```
## S4 method for signature 'kRp.corpus'
read.corp.custom(corpus, caseSens = TRUE, log.base = 10,
      keep_dtm = FALSE, ...)
```

Arguments

`corpus`	An object of class `kRp.corpus`.
`caseSens`	Logical. If `FALSE`, all tokens will be matched in their lower case form.
`log.base`	A numeric value defining the base of the logarithm used for inverse document frequency (idf). See `log` for details.
`keep_dtm`	Logical. If `TRUE` and `corpus` does not yet provide a `document term matrix`, the one generated during calculation will be added to the resulting object.
`...`	Options to pass through to the `read.corp.custom` method for objects of the class union `kRp.text`.

Details

Since the analysis is based on a document term matrix, a pre-existing matrix as a feature of the `corpus` object will be used if it matches the case sensitivity setting. Otherwise a new matrix will be generated (but not replace the existing one). If no document term matrix is present yet, also one will be generated and can be kept as an additional feature of the resulting object.

Value

An object of the same class as `corpus`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
```

```
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  myCorpus <- read.corp.custom(myCorpus)
  corpusCorpFreq(myCorpus)
} else {}
```

`readability,kRp.corpus-method`

Apply readability() to all texts in kRp.corpus objects

Description

This method calls `readability` on all tagged text objects inside the given `txt.file` object (using `mclapply`).

Usage

```
## S4 method for signature 'kRp.corpus'
readability(
  txt.file,
  summary = TRUE,
  mc.cores = getOption("mc.cores", 1L),
  quiet = TRUE,
  ...
)
```

Arguments

`txt.file`	An object of class `kRp.corpus`.
`summary`	Logical, determines if the `summary` slot should automatically be updated by calling `summary` on the result.
`mc.cores`	The number of cores to use for parallelization, see `mclapply`.
`quiet`	Logical, if `FALSE` shows a status bar for some calculations of each text, see `readability` for details.
`...`	options to pass through to `readability`.

Value

An object of the same class as `txt.file`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  myTexts <- readability(myCorpus)
  corpusSummary(myCorpus)
} else {}
```

---

readCorpus *Create kRp.corpus objects from text files or data frames*

Description

You can either read a corpus from text files (one file per text, also see the Hierarchy section below) or from TIF compliant data frames (see the Data frames section below).

Usage

```
readCorpus(
  dir,
  hierarchy = list(),
  lang = "kRp.env",
  tagger = "kRp.env",
  encoding = "",
  pattern = NULL,
  recursive = FALSE,
  ignore.case = FALSE,
  mode = "text",
  format = "file",
  mc.cores = getOption("mc.cores", 1L),
  id = "",
  ...
)
```

Arguments

dir	Either a file path to the root directory of the text corpus, or a TIF compliant data frame. If a directory path (character string), texts can be recursively ordered into subfolders named exactly as defined by hierarchy. If hierarchy is an empty list, all text files located in dir are parsed without a hierachical structure. If a data frame, also set format="obj" and provide hierarchy levels as additional columns, as described in the Data frames section.
hierarchy	A named list of named character vectors describing the directory hierarchy level by level. If TRUE instead, the hierarchy structure is taken directly from the directory tree. See section Hierarchy for details.
lang	A character string naming the language of the analyzed corpus. See kRp.POS.tags for all supported languages. If set to "kRp.env" this is got from get.kRp.env. This information will also be passed to the readerControl list of the VCorpus call.
tagger	A character string pointing to the tokenizer/tagger command you want to use for basic text analysis. Defaults to tagger="kRp.env" to get the settings by get.kRp.env. Set to "tokenize" to use tokenize.
encoding	Character string describing the current encoding. See DirSource for details, omitted if format="obj".
pattern	A regular expression for file matching. See DirSource for details, omitted if format="obj".
recursive	Logical, indicates whether directories should be read recursively. See DirSource for details, omitted if format="obj".
ignore.case	Logical, indicates whether pattern is matched case sensitive. See DirSource for details, omitted if format="obj".
mode	Character string defining the reading mode. See DirSource for details, omitted if format="obj".
format	Either "file" or "obj", depending on whether you want to scan files or analyze the text in a given object, like a character vector. If the latter and treetag is used as the tagger, texts will be written to temporary files for the process (see dir).
mc.cores	The number of cores to use for parallelization, see mclapply. This value is passed through to simpleCorpus.
id	A character string describing the main subject/purpose of the text corpus.
...	Additional options which are passed through to the defined tagger.

Value

An object of class kRp.corpus.

Hierarchy

To import a hierarchically structured text corpus you must categorize all texts in a directory structure that resembles the hierarchy. If for example you would like to import a corpus on two different topics and two differnt sources, your hierarchy has two nested levels (topic and source). The root

directory `dir` would then need to have two subdirectories (one for each topic) which in turn must have two subdirectories (one for each source), and the actual text files are found in those.

To use this hierarchical structure in `readCorpus`, the `hierarchy` argument is used. It is a named list, where each list item represents one hierachical level (here again topic and source), and its value is a named character vector describing the actual topics and sources to be used. It is important to understand how these character vectors are treated: The names of elements must exactly match the corresponding subdirectroy name, whereas the value is a free text description. The names of the list items however describe the hierachical level and are not matched with directory names.

Data frames

In order to import a corpus from a data frame, the object must be in Text Interchange Format (TIF) as described by [1]. As a minimum, the data frame must have two character columns, `doc_id` and `text`.

You can provide additional information on hierarchical categories by using further columns, where the column name must match the category name (hierachical level). The order of those columns in the data frame is not important, as you must still fully define the hierarchical structure using the `hierarchy` argument. All columns you omit are ignored, but the values used in the `hierarchy` list and the respective columns must match, as rows with unmatched category levels will also be ignored.

Note that the special column names `path` and `file` will also be imported automatically.

References

[1] Text Interchange Formats (https://github.com/ropensci/tif)

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  # "flat" corpus, parse all texts in the given dir
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Winner", "Wikipedia_prev"
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  # corpus with one category names "Source"
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Winner"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
```

```
    )
  ),
  tagger="tokenize",
  lang="en"
)

# two hieraryhical levels, "Topic" and "Source"
myCorpus <- readCorpus(
  dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
  hierarchy=list(
    Topic=c(
      Winner="Reality Winner",
      Edwards="Natalie Edwards"
    ),
    Source=c(
      Wikipedia_prev="Wikipedia (old)",
      Wikipedia_new="Wikipedia (new)"
    )
  ),
  tagger="tokenize",
  lang="en"
)

# get hierarchy from directory tree
myCorpus <- readCorpus(
  dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
  hierarchy=TRUE,
  tagger="tokenize",
  lang="en"
)

## Not run:
  # if the same corpus is available as TIF compliant data frame
  myCorpus <- readCorpus(
    dir=myCorpus_df,
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    lang="en",
    format="obj"
  )

## End(Not run)
} else {}
```

show,kRp.corpus-method

Show methods for kRp.corpus objects

Description

Show methods for S4 objects of class kRp.corpus.

Usage

```
## S4 method for signature 'kRp.corpus'
show(object)
```

Arguments

object	An object of class kRp.corpus.

split_by_doc_id,kRp.corpus-method

Turn a kRp.corpus object into a list of kRp.text objects

Description

For some analysis steps it might be important to have individual tagged texts instead of one large corpus object. This method produces just that.

Usage

```
## S4 method for signature 'kRp.corpus'
split_by_doc_id(obj, keepFeatures = TRUE)
```

Arguments

obj	An object of class kRp.corpus.
keepFeatures	Either logical, whether to keep all features or drop them, or a character vector of names of features to keep if present.

Value

A named list of objects of class kRp.text. Elements are named by their doc_id.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(path.package("tm.plugin.koRpus"), "examples", "corpus"),
    hierarchy=list(
      Topic=c(
        Winner="Reality Winner",
        Edwards="Natalie Edwards"
      ),
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  myCorpusList <- split_by_doc_id(myCorpus)
} else {}
```

---

summary,kRp.corpus-method

Apply summary() to all texts in kRp.corpus objects

Description

This method performs a summary call on all text objects inside the given object object. Contrary to what other summary methods do, this method always returns the full object with an updated summary slot.

Usage

```
## S4 method for signature 'kRp.corpus'
summary(object, missing = NA, ...)

corpusSummary(obj)

## S4 method for signature 'kRp.corpus'
corpusSummary(obj)

corpusSummary(obj) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusSummary(obj) <- value
```

Arguments

object	An object of class kRp.corpus.
missing	Character string to use for missing values.
...	Used for internal processes.
obj	An object of class kRp.corpus.
value	The new value to replace the current with.

Details

The summary slot contains a data.frame with aggregated information of all texts that the respective object contains.

corpusSummary is a simple method to get or set the summary slot in kRp.corpus objects directly.

Value

An object of the same class as object.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  # calculate readability, but prevent a summary table from being added
  myCorpus <- readability(myCorpus, summary=FALSE)
  corpusSummary(myCorpus)

  # add summaries
  myCorpus <- summary(myCorpus)
  corpusSummary(myCorpus)
} else {}
```

taggedText,kRp.corpus-method

Getter/setter methods for kRp.corpus objects

Description

These methods should be used to get or set values of text objects generated by functions like readCorpus.

Usage

```
## S4 method for signature 'kRp.corpus'
taggedText(obj)

## S4 replacement method for signature 'kRp.corpus'
taggedText(obj) <- value

## S4 method for signature 'kRp.corpus'
doc_id(obj, has_id = NULL)

## S4 method for signature 'kRp.corpus'
describe(obj, doc_id = NULL, simplify = TRUE, ...)

## S4 replacement method for signature 'kRp.corpus'
describe(obj, doc_id = NULL, ...) <- value

## S4 method for signature 'kRp.corpus'
language(obj)

## S4 replacement method for signature 'kRp.corpus'
language(obj) <- value

## S4 method for signature 'kRp.corpus'
hasFeature(obj, feature = NULL)

## S4 replacement method for signature 'kRp.corpus'
hasFeature(obj, feature) <- value

## S4 method for signature 'kRp.corpus'
feature(obj, feature, doc_id = NULL)

## S4 replacement method for signature 'kRp.corpus'
feature(obj, feature) <- value

## S4 method for signature 'kRp.corpus'
corpusReadability(obj, doc_id = NULL)
```

```
## S4 replacement method for signature 'kRp.corpus'
corpusReadability(obj) <- value

corpusTm(obj)

## S4 method for signature 'kRp.corpus'
corpusTm(obj)

corpusTm(obj) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusTm(obj) <- value

corpusMeta(obj, meta = NULL, fail = TRUE)

## S4 method for signature 'kRp.corpus'
corpusMeta(obj, meta = NULL, fail = TRUE)

corpusMeta(obj, meta = NULL) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusMeta(obj, meta = NULL) <- value

## S4 method for signature 'kRp.corpus'
corpusHyphen(obj, doc_id = NULL)

## S4 replacement method for signature 'kRp.corpus'
corpusHyphen(obj) <- value

## S4 method for signature 'kRp.corpus'
corpusLexDiv(obj, doc_id = NULL)

## S4 replacement method for signature 'kRp.corpus'
corpusLexDiv(obj) <- value

## S4 method for signature 'kRp.corpus'
corpusFreq(obj)

## S4 replacement method for signature 'kRp.corpus'
corpusFreq(obj) <- value

## S4 method for signature 'kRp.corpus'
corpusCorpFreq(obj)

## S4 replacement method for signature 'kRp.corpus'
corpusCorpFreq(obj) <- value

corpusHierarchy(obj, ...)
```

```
## S4 method for signature 'kRp.corpus'
corpusHierarchy(obj)

corpusHierarchy(obj) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusHierarchy(obj) <- value

corpusFiles(obj, paths = FALSE, ...)

## S4 method for signature 'kRp.corpus'
corpusFiles(obj, paths = FALSE)

corpusFiles(obj) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusFiles(obj) <- value

corpusDocTermMatrix(obj, ...)

## S4 method for signature 'kRp.corpus'
corpusDocTermMatrix(obj)

corpusDocTermMatrix(obj, terms = NULL, case.sens = NULL, tfidf = NULL) <- value

## S4 replacement method for signature 'kRp.corpus'
corpusDocTermMatrix(obj, terms = NULL, case.sens = NULL,
      tfidf = NULL) <- value

## S4 method for signature 'kRp.corpus'
corpusStopwords(obj)

## S4 replacement method for signature 'kRp.corpus'
corpusStopwords(obj) <- value

## S4 method for signature 'kRp.corpus'
diffText(obj, doc_id = NULL)

## S4 replacement method for signature 'kRp.corpus'
diffText(obj) <- value

## S4 method for signature 'kRp.corpus'
originalText(obj)

is.corpus(obj)

## S4 method for signature 'kRp.corpus,ANY,ANY,ANY'
```

```
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'kRp.corpus,ANY,ANY,ANY'
x[i, j, ...] <- value

## S4 method for signature 'kRp.corpus'
x[[i, doc_id = NULL, ...]]

## S4 replacement method for signature 'kRp.corpus'
x[[i, doc_id = NULL, ...]] <- value

## S4 method for signature 'kRp.corpus'
tif_as_tokens_df(tokens)

tif_as_corpus_df(corpus)

## S4 method for signature 'kRp.corpus'
tif_as_corpus_df(corpus)
```

Arguments

obj	An object of class kRp.corpus.
value	A new value to replace the current with.
has_id	A character vector with doc_ids to look for in the object. The return value is then a logical vector of the same length, indicating if the respective id was found or not.
doc_id	A character vector to limit the scope to one or more particular document IDs.
simplify	If TRUE and result is a list of length 1, return the list element.
...	Additional arguments to pass through, depending on the method.
feature	Character string naming the object feature to look for.
meta	If not NULL, the meta list entry of the given name.
fail	Logical, whether the method should fail with an error if meta was not found. If set to FALSE, returns invisible(NULL) instead.
paths	Logical, indicates for corpusFiles() whether full paths should be returned, or just the actual file name.
terms	A character string defining the tokens used for calculating the matrix. Stored in object's meta data slot.
case.sens	Logical, whether terms were counted case sensitive. Stored in object's meta data slot.
tfidf	Logical, use TRUE if the term frequency–inverse document frequency (tf-idf) values were calculated instead of absolute frequency. Stored in object's meta data slot.
x	See obj.
i	Defines the row selector ([) or the name to match ([[) in the tokens slot.

j	Defines the column selector in the tokens slot.
drop	See [.
tokens	An object of class kRp.corpus.
corpus	An object of class kRp.corpus.

Details

- taggedText() returns the tokens slot.
- describe() returns the desc slot.
- hasFeature() returns TRUE or codeFALSE, depending on whether the requested feature is present or not.
- feature() returns the list entry of the feat_list slot for the requested feature.
- corpusReadability() returns the list of kRp.readability objects.
- corpusTm() returns the VCorpus object.
- corpusMeta() returns the list with meta information.
- corpusHyphen() returns the list of kRp.hyphen objects.
- corpusLexDiv() returns the list of kRp.TTR objects.
- corpusFiles() returns the character vector of file names of the object.
- corpusFreq() returns the frequency analysis data from the feat_list slot.
- corpusCorpFreq() returns the kRp.corp.freq object of the feat_list slot.
- corpusHierarchy() returns the corpus' hierarchy structure.
- corpusDocTermMatrix() returns the sparse document term matrix of the feat_list slot.
- corpusStopwords() returns the number of stopwords found in each text (if analyzed) from the feat_list slot.
- diffText() returns the diff element of the feat_list slot.
- originalText regenerates the original text before text transformations and returns it as a data frame.
- [/[[can be used as a shortcut to index the results of taggedText().
- tif_as_corpus_df returns the whole corpus in a single TIF[1] compliant data.frame.
- tif_as_tokens_df returns the tokens slot in a TIF[1] compliant data.frame, i.e., doc_id is not a factor but a character vector.

References

[1] Text Interchange Formats (https://github.com/ropensci/tif)

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Winner", "Wikipedia_new"
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  taggedText(myCorpus)

  corpusMeta(myCorpus, "note") <- "an interesting read!"

  # export object to TIF compliant data frame
  myCorpus_df <- tif_as_corpus_df(myCorpus)
} else {}
```

---

textTransform,kRp.corpus-method

Apply textTransform() to all texts in kRp.corpus objects

Description

This method calls `textTransform` on all tagged text objects inside the given `txt` object (using `mclapply`).

Usage

```
## S4 method for signature 'kRp.corpus'
textTransform(txt, mc.cores = getOption("mc.cores", 1L), ...)
```

Arguments

`txt`	An object of class `kRp.corpus`.
`mc.cores`	The number of cores to use for parallelization, see `mclapply`.
`...`	options to pass through to `textTransform`.

Value

An object of the same class as `txt`.

Examples

```
# use readCorpus() to create an object of class kRp.corpus
# code is only run when the english language package can be loaded
if(require("koRpus.lang.en", quietly = TRUE)){
  myCorpus <- readCorpus(
    dir=file.path(
      path.package("tm.plugin.koRpus"), "examples", "corpus", "Edwards"
    ),
    hierarchy=list(
      Source=c(
        Wikipedia_prev="Wikipedia (old)",
        Wikipedia_new="Wikipedia (new)"
      )
    ),
    # use tokenize() so examples run without a TreeTagger installation
    tagger="tokenize",
    lang="en"
  )

  head(taggedText(myCorpus), n=10)
  myCorpus <- textTransform(myCorpus, scheme="minor")
  head(taggedText(myCorpus), n=10)
} else {}
```

Index