Package 'tm'

January 13, 2014

Title Text Mining Package

Version 0.5-10

Date 2014-01-07

Depends R (>= 3.0.0)

Imports parallel, slam (>= 0.1-31)

Suggests filehash, proxy, Rcampdf, Rgraphviz, Rpoppler, SnowballC, XML

SystemRequirements Antiword (http://www.winfield.demon.nl/) for reading MS Word files, pdfinfo and pdftotext from Poppler (http://poppler.freedesktop.org/) for reading PDF

Description A framework for text mining applications within R.

License GPL-3

URL http://tm.r-forge.r-project.org/

Author Ingo Feinerer [aut, cre],Kurt Hornik [aut],Artifex Software, Inc. [ctb, cph] (pdf_info.ps taken from GPL Ghostscript)

Maintainer Ingo Feinerer <feinerer@logic.at>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-01-13 18:40:58

R topics documented:

---

acq *50 Exemplary News Articles from the Reuters-21578 XML Data Set of Topic acq*

Description

This dataset holds 50 news articles with additional meta information from the Reuters-21578 XML data set. All documents belong to the topic acq dealing with corporate acquisitions.

Usage

```
data("acq")
```

Format

A corpus of 50 text documents.

Source

Reuters-21578 Text Categorization Collection Distribution 1.0 (XML format).

References

Lewis, David (1997) *Reuters-21578 Text Categorization Collection Distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Luz, Saturnino XML-*encoded version of Reuters-21578*. http://modnlp.berlios.de/reuters21578.html

Examples

```
data("acq")
summary(acq)
```

`as.PlainTextDocument` *Create Objects of Class PlainTextDocument*

Description

Create objects of class PlainTextDocument.

Usage

```
## S3 method for class 'PlainTextDocument'
as.PlainTextDocument(x)
## S3 method for class 'Reuters21578Document'
as.PlainTextDocument(x)
## S3 method for class 'RCV1Document'
as.PlainTextDocument(x)
```

Arguments

x	A text document.

See Also

`getTransformations` to list available transformation (mapping) functions.

Examples

```
reut21578 <- system.file("texts", "crude", package = "tm")
r <- Corpus(DirSource(reut21578),
                      readerControl = list(reader = readReut21578XML))
r[[1]]
as.PlainTextDocument(r[[1]])
```

crude — *20 Exemplary News Articles from the Reuters-21578 XML Data Set of Topic crude*

Description

This data set holds 20 news articles with additional meta information from the Reuters-21578 XML data set. All documents belong to the topic crude dealing with crude oil.

Usage

```
data("crude")
```

Format

A corpus of 20 text documents.

Source

Reuters-21578 Text Categorization Collection Distribution 1.0 (XML format).

References

Lewis, David (1997) *Reuters-21578 Text Categorization Collection Distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Luz, Saturnino XML-*encoded version of Reuters-21578*. http://modnlp.berlios.de/reuters21578.html

Examples

```
data("crude")
summary(crude)
```

DataframeSource — *Data Frame Source*

Description

Constructs a source from a data frame.

Usage

```
DataframeSource(x, encoding = "unknown")
```

Arguments

x	A data frame holding the texts.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Value

An object of class DataframeSource which extends the class Source representing a data frame interpreting each row as a document.

Author(s)

Ingo Feinerer

See Also

getSources to list available sources. Encoding on encodings in R.

Examples

```
docs <- data.frame(docs = c("This is a text.", "This another one."),
                   row.names = c("Text 1", "Text 2"))
(ds <- DataframeSource(docs))
inspect(Corpus(ds))
```

DirSource *Directory Source*

Description

Constructs a directory source.

Usage

```
DirSource(directory = ".", encoding = "unknown", pattern = NULL,
          recursive = FALSE, ignore.case = FALSE)
```

Arguments

directory	A character vector of full path names; the default corresponds to the working directory getwd().
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.
pattern	An optional regular expression. Only file names which match the regular expression will be returned.
recursive	Logical. Should the listing recurse into directories?
ignore.case	Logical. Should pattern-matching be case-insensitive?

Value

An object of class `DirSource` which extends the class `Source` representing a directory. Each file in this directory is considered to be a document.

Author(s)

Ingo Feinerer

See Also

`getSources` to list available sources. `Encoding` on encodings in R.

Examples

```
DirSource(system.file("texts", "txt", package = "tm"))
```

---

dissimilarity *Dissimilarity*

Description

Compute the dissimilarity between documents in a term-document matrix or between two explicit documents.

Usage

```
## S3 method for class 'TermDocumentMatrix'
dissimilarity(x, y = NULL, method)
## S3 method for class 'PlainTextDocument'
dissimilarity(x, y = NULL, method)
```

Arguments

`x`	Either a term-document matrix or a text document.
`y`	A text document. Only used if `x` is a text document.
`method`	Dissimilarity measure. Any method accepted by `dist` from package **proxy** can be passed over.

Value

An object of class `dist` representing the dissimilarity between participating documents.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)
dissimilarity(tdm, method = "cosine")
dissimilarity(crude[[1]], crude[[2]], method = "eJaccard")
```

findAssocs *Find Associations in a Term-Document Matrix*

Description

Find associations in a document-term or term-document matrix.

Usage

```
## S3 method for class 'DocumentTermMatrix'
findAssocs(x, terms, corlimit)
## S3 method for class 'TermDocumentMatrix'
findAssocs(x, terms, corlimit)
```

Arguments

x	A DocumentTermMatrix or a TermDocumentMatrix.
terms	a character vector holding terms.
corlimit	a numeric vector (of the same length as terms; recycled otherwise) for the (inclusive) lower correlation limits of each term in the range from zero to one.

Value

A named list. Each list component is named after a term in terms and contains a named numeric vector. Each vector holds matching terms from x and their rounded correlations satisfying the inclusive lower correlation limit of corlimit.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)
findAssocs(tdm, c("oil", "opec", "xyz"), c(0.7, 0.75, 0.1))
```

findFreqTerms *Find Frequent Terms*

Description

Find frequent terms in a document-term or term-document matrix.

Usage

```
findFreqTerms(x, lowfreq = 0, highfreq = Inf)
```

Arguments

x	A DocumentTermMatrix or TermDocumentMatrix.
lowfreq	A numeric for the lower frequency bound.
highfreq	A numeric for the upper frequency bound.

Details

This method works for all numeric weightings but is probably most meaningful for the standard term frequency (tf) weighting of x.

Value

A character vector of terms in x which occur more or equal often than lowfreq times and less or equal often than highfreq times.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)
findFreqTerms(tdm, 2, 3)
```

---

foreign *Read Document-Term Matrices*

Description

Read document-term matrices stored in special file formats.

Usage

```
read_dtm_Blei_et_al(file, vocab = NULL)
read_dtm_MC(file, scalingtype = NULL)
```

Arguments

file	a character string with the name of the file to read.
vocab	a character string with the name of a vocabulary file (giving the terms, one per line), or NULL.
scalingtype	a character string specifying the type of scaling to be used, or NULL (default), in which case the scaling will be inferred from the names of the files with non-zero entries found (see **Details**).

Details

`read_dtm_Blei_et_al` reads the (List of Lists type sparse matrix) format employed by the Latent Dirichlet Allocation and Correlated Topic Model C codes by Blei et al (http://www.cs.princeton.edu/~blei).

MC is a toolkit for creating vector models from text documents (see http://www.cs.utexas.edu/users/dml/software/mc/). It employs a variant of Compressed Column Storage (CCS) sparse matrix format, writing data into several files with suitable names: e.g., a file with '`_dim`' appended to the base file name stores the matrix dimensions. The non-zero entries are stored in a file the name of which indicates the scaling type used: e.g., '`_tfx_nz`' indicates scaling by term frequency ('`t`'), inverse document frequency ('`f`') and no normalization ('`x`'). See '`README`' in the MC sources for more information.

`read_dtm_MC` reads such sparse matrix information with argument `file` giving the path with the base file name.

Value

A document-term matrix.

See Also

`read_stm_MC` in package **slam**.

`FunctionGenerator` *Function Generator*

Description

Construct a function generator.

Usage

```
FunctionGenerator(x)
```

Arguments

`x`	A generator function which takes some input and constructs and returns a new function based on that input information.

Value

An object of class `FunctionGenerator` which extends the class `function` representing a function generator.

Author(s)

Ingo Feinerer

See Also

Most reader functions (use `getReaders` to list available readers) are function generators.

Examples

```
funGen <- FunctionGenerator(function(y, ...) {
if (is(y, "integer")) function(x) x+1 else function(x) x-1
})
funGen
funGen(3L)
funGen("a")
```

`getReaders` *List Available Readers*

Description

List available readers.

Usage

```
getReaders()
```

Value

A character vector with available readers.

Author(s)

Ingo Feinerer

See Also

`readDOC`, `readPDF`, `readPlain`, `readRCV1`, `readReut21578XML`, `readTabular`, `readXML`.

Examples

```
getReaders()
```

getSources *List Available Sources*

Description

List available sources.

Usage

```
getSources()
```

Value

A character vector with available sources.

Author(s)

Ingo Feinerer

See Also

DataframeSource, DirSource, ReutersSource, URISource, VectorSource.

Examples

```
getSources()
```

getTokenizers *List Available Tokenizers*

Description

List available tokenizers.

Usage

```
getTokenizers()
```

Value

A character vector with available tokenizers.

Author(s)

Ingo Feinerer

See Also

MC_tokenizer and scan_tokenizer.

Examples

```
getTokenizers()
```

---

getTransformations *List Available Transformations*

Description

List available transformations (mappings) which can be used with tm_map.

Usage

```
getTransformations()
```

Value

A character vector with available transformations.

Author(s)

Ingo Feinerer

See Also

as.PlainTextDocument, removeNumbers, removePunctuation, removeWords, stemDocument, stripWhitespace.

Examples

```
getTransformations()
```

inspect *Inspect Objects*

Description

Inspect, i.e., display detailed information on a corpus or a term-document matrix.

Usage

```
## S3 method for class 'PCorpus'
inspect(x)
## S3 method for class 'VCorpus'
inspect(x)
## S3 method for class 'TermDocumentMatrix'
inspect(x)
```

Arguments

x	Either a corpus or a term-document matrix.

Examples

```
data("crude")
inspect(crude[1:3])
tdm <- TermDocumentMatrix(crude)[1:10, 1:10]
inspect(tdm)
```

makeChunks *Split a Corpus into Chunks*

Description

Split a corpus into equally sized chunks conserving document boundaries.

Usage

```
makeChunks(corpus, chunksize)
```

Arguments

corpus	The corpus to be split into chunks.
chunksize	The chunk size.

Value

A corpus consisting of the chunks. Note that corpus meta data is not passed on to the newly created chunk corpus.

Author(s)

Ingo Feinerer

Examples

```
txt <- system.file("texts", "txt", package = "tm")
ovid <- Corpus(DirSource(txt))
sapply(ovid, length)
ovidChunks <- makeChunks(ovid, 5)
sapply(ovidChunks, length)
```

---

materialize *Materialize Lazy Mappings*

Description

The function tm_map supports so-called lazy mappings, that are mappings which are delayed until the documents' content is accessed. This function triggers the evaluation, i.e., it materializes the documents.

Usage

```
materialize(corpus, range = seq_along(corpus))
```

Arguments

corpus	A document collection with lazy mappings.
range	The indices of documents to be materialized.

Value

A corpus with materialized, i.e., all mappings computed and applied, documents for the requested range.

Author(s)

Ingo Feinerer

See Also

tm_map

Examples

```
data("crude")
x <- tm_map(crude, stemDocument, lazy = TRUE)
x <- materialize(x)
```

meta *Meta Data Management*

Description

Methods to access and modify meta data of documents, corpora, and repositories. In addition to **tm**'s internal meta data structures, Simple Dublin Core meta data mappings are available.

Usage

```
## S3 method for class 'Corpus'
meta(x, tag, type = c("indexed", "corpus", "local"))
## S3 method for class 'TextDocument'
meta(x, tag, type = NULL)
## S3 method for class 'TextRepository'
meta(x, tag, type = NULL)
content_meta(x, tag) <- value
DublinCore(x, tag = NULL)
```

Arguments

x	Either a text document, a corpus, or a text repository.
tag	A character identifying the name of the meta datum.
type	A character specifying which meta data of a corpus should be considered.
value	Replacement value.

Details

In general this function can be used to display meta information but also to modify individual meta data:

x = "TextDocument", tag = NULL If no tag is given, this method pretty prints all x's meta data. If tag is provided its value in the meta data is returned.

x = "Corpus", tag = NULL, type = "indexed" This method investigates the type argument. type must be either indexed (default), local, or corpus. Former is a shortcut for accessing document level meta data (DMetaData) stored at the collection level (because it forms an own entity, or for performance reasons, i.e., a form of indexing, hence the name indexed), local accesses the meta data local to each text document (i.e., meta data in text documents' attributes), and corpus is a shortcut for corpus specific meta data (CMetaData). Depending whether a tag is set or not, all or only the meta data identified by the tag is displayed or modified.

x = "TextRepository", tag = NULL If no tag is given, this method pretty prints all x's meta data. If tag is provided its value in the meta data is returned.

Simple Dublin Core meta data is only available locally at each document:

x = "TextDocument", tag = NULL Returns or sets the Simple Dublin Core meta datum named `tag` for `x`. `tag` must be a valid Simple Dublin Core element name (i.e, title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, or rights) or `NULL`. For the latter all Dublin Core meta data are printed.

`content_meta` is a convenience wrapper which calls `Content` if `tag = "Content"` and `meta` otherwise.

References

Dublin Core Metadata Initiative. http://dublincore.org/

Examples

```
data("crude")
meta(crude[[1]])
DublinCore(crude[[1]])
meta(crude[[1]], tag = "Topics")
meta(crude[[1]], tag = "Comment") <- "A short comment."
meta(crude[[1]], tag = "Topics") <- NULL
DublinCore(crude[[1]], tag = "creator") <- "Ano Nymous"
DublinCore(crude[[1]], tag = "Format") <- "XML"
DublinCore(crude[[1]])
meta(crude[[1]])
meta(crude)
meta(crude, type = "corpus")
meta(crude, "labels") <- 21:40
meta(crude)
```

---

names *Row, Column, Dim Names, Document IDs, and Terms*

Description

Retrieve the row names, column names, dimnames, document IDs, and terms of a term-document matrix or document-term matrix.

Arguments

x	Either a term-document matrix or document-term matrix.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)[1:10,1:20]
rownames(tdm)
colnames(tdm)
dimnames(tdm)
Docs(tdm)
Terms(tdm)
```

number — *The Number of Rows/Columns/Dimensions/Documents/Terms of a Term-Document Matrix*

Description

Return the number of rows, columns, dimensions, documents, and terms of a term-document matrix or a document-term matrix.

Arguments

x	Either a term-document matrix or a document-term matrix.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)[1:10,1:20]
ncol(tdm)
nrow(tdm)
dim(tdm)
nDocs(tdm)
nTerms(tdm)
```

PCorpus — *Permanent Corpus Constructor*

Description

Construct a permanent corpus.

Usage

```
PCorpus(x,
        readerControl = list(reader = x$DefaultReader, language = "en"),
        dbControl = list(dbName = "", dbType = "DB1"))
DBControl(x)
## S3 method for class 'PCorpus'
DMetaData(x)
```

Arguments

x	A Source object for PCorpus, and a corpus for the other functions.
readerControl	A list with the named components reader representing a reading function capable of handling the file format found in x, and language giving the text's language (preferably as IETF language tags). The default language is assumed to be English ("en"). Use NA to avoid internal assumptions (e.g., when the language is unknown or is deliberately not set).

`dbControl` A list with the named components `dbName` giving the filename holding the sourced out documents (i.e., the database), and `dbType` holding a valid database type as supported by package **filehash**. Under activated database support the `tm` package tries to keep as few as possible resources in memory under usage of the database.

Details

Permanent means that documents are physically stored outside of R (e.g., in a database) and R objects are only pointers to external structures. I.e., changes in the underlying external representation can affect multiple R objects simultaneously.

The constructed corpus object inherits from a `list` and has three attributes containing meta and database management information:

`CMetaData` Corpus Meta Data contains corpus specific meta data in form of tag-value pairs and information about children in form of a binary tree. This information is useful for reconstructing meta data after e.g. merging corpora.

`DMetaData` Document Meta Data of class `data.frame` contains document specific meta data for the corpus. This data frame typically encompasses clustering or classification results which basically are metadata for documents but form an own entity (e.g., with its name, the value range, etc.).

`DBControl` Database control field is a `list` with two named components: `dbName` holds the path to the permanent database storage, and `dbType` stores the database type.

Value

An object of class `PCorpus` which extends the classes `Corpus` and `list` containing a permanent corpus.

Author(s)

Ingo Feinerer

Examples

```
txt <- system.file("texts", "txt", package = "tm")
## Not run: PCorpus(DirSource(txt),
        dbControl = list(dbName = "myDB.db", dbType = "DB1"))
## End(Not run)
```

PlainTextDocument *Plain Text Document*

Description

Construct an object representing a plain text document with additional meta information.

Usage

```
PlainTextDocument(x = character(0), author = character(0),
                  datetimestamp = as.POSIXlt(Sys.time(), tz = "GMT"),
                  description = character(0), heading = character(0),
                  id = character(0), origin = character(0),
                  language = character(0), localmetadata = list())
```

Arguments

x	Object of class character containing the content.
author	Object of class character containing the author names.
datetimestamp	Object of class POSIXlt containing the date and time when the document was written.
description	Object of class character containing additional text information.
heading	Object of class character containing the title or a short heading.
id	Object of class character containing an identifier.
origin	Object of class character containing information on the source and origin of the text.
language	Object of class character containing the language of the text (preferably as IETF language tags).
localmetadata	Object of class list containing local meta data in form of tag-value pairs.

Author(s)

Ingo Feinerer

---

plot *Visualize a Term-Document Matrix*

Description

Visualize correlations between terms of a term-document matrix.

Usage

```
## S3 method for class 'TermDocumentMatrix'
plot(x,
     terms = sample(Terms(x), 20),
     corThreshold = 0.7,
     weighting = FALSE,
     attrs = list(graph = list(rankdir = "BT"),
                  node = list(shape = "rectangle",
                              fixedsize = FALSE)),
     ...)
```

Arguments

x	A term-document matrix.
terms	Terms to be plotted. Defaults to 20 randomly chosen terms of the term-document matrix.
corThreshold	Do not plot correlations below this threshold. Defaults to 0.7.
weighting	Define whether the line width corresponds to the correlation.
attrs	Argument passed to the plot method for class graphNEL.
...	Other arguments passed to the graphNEL plot method.

Details

Visualization requires that package **Rgraphviz** is available.

Examples

```
## Not run: data(crude)
tdm <- TermDocumentMatrix(crude,
                          control = list(removePunctuation = TRUE,
                                         removeNumbers = TRUE,
                                         stopwords = TRUE))
plot(tdm, corThreshold = 0.2, weighting = TRUE)
## End(Not run)
```

---

prescindMeta *Prescind Document Meta Data*

Description

Extracts meta data from each individual document (either stored in its attributes or in additional user-defined local meta data pairs) of a corpus and creates a data frame which contains both the global meta data information of the corpus plus the extracted (i.e., shifted up) local meta data of the individual text documents.

Usage

```
prescindMeta(x, meta)
```

Arguments

x	A corpus.
meta	A character vector of meta data names to be shifted up.

Value

A data frame constructed from x with shifted up meta data.

See Also

DMetaData, and meta

Examples

```
data("crude")
DMetaData(crude)
meta(crude, tag = "ID", type = "local")
prescindMeta(crude, c("ID", "Heading"))
```

RCV1Document *RCV1 Text Document*

Description

Construct an object representing a RCV1 XML text document with meta information.

Usage

```
RCV1Document(x, author = character(0),
             datetimestamp = as.POSIXlt(Sys.time(), tz = "GMT"),
             description = character(0), heading = character(0),
             id = character(0), origin = character(0),
             language = character(0), localmetadata = list())
```

Arguments

x	Object of class list containing the content.
author	Object of class character containing the author names.
datetimestamp	Object of class POSIXlt containing the date and time when the document was written.
description	Object of class character containing additional text information.
heading	Object of class character containing the title or a short heading.
id	Object of class character containing an identifier.
origin	Object of class character containing information on the source and origin of the text.
language	Object of class character containing the language of the text (preferably as IETF language tags).
localmetadata	Object of class list containing local meta data in form of tag-value pairs.

Author(s)

Ingo Feinerer

References

Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, **5**, 361–397. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf

See Also

PlainTextDocument and Reuters21578Document

---

readDOC *Read In a MS Word Document*

Description

Return a function which reads in a Microsoft Word document extracting its text.

Usage

```
readDOC(AntiwordOptions = "")
```

Arguments

AntiwordOptions
: Options passed over to antiword.

Details

Formally this function is a function generator, i.e., it returns a function (which reads in a text document) with a well-defined signature, but can access passed over arguments (e.g., options to antiword) via lexical scoping.

Note that this MS Word reader needs the tool antiword installed and accessible on your system. This can convert documents from Microsoft Word version 2, 6, 7, 97, 2000, 2002 and 2003 to plain text, and is available from http://www.winfield.demon.nl/.

Value

A function with the signature elem, language, id:

elem a list with the named component uri which must hold a valid file name.

language a string giving the text's language.

id a unique identification string for the returned text document.

The function returns a PlainTextDocument representing the text and meta data extracted from elem$uri.

Author(s)

Ingo Feinerer

See Also

getReaders to list available reader functions.

---

readPDF *Read In a PDF Document*

Description

Return a function which reads in a portable document format (PDF) document extracting both its text and its meta data.

Usage

```
readPDF(engine = c("xpdf", "Rpoppler", "ghostscript", "Rcampdf", "custom"),
        control = list(info = NULL, text = NULL))
```

Arguments

engine	a character string for the preferred PDF extraction engine (see **Details**).
control	a list of control options for the engine with the named components info and text (see **Details**).

Details

Formally this function is a function generator, i.e., it returns a function (which reads in a text document) with a well-defined signature, but can access passed over arguments (e.g., the preferred PDF extraction engine and control options) via lexical scoping.

Available PDF extraction engines are as follows.

"xpdf" (default) command line pdfinfo and pdftotext executables which must be installed and accessible on your system. Suitable utilities are provided by the Xpdf (http://www.foolabs.com/xpdf/) PDF viewer or by the Poppler (http://poppler.freedesktop.org/) PDF rendering library.

"Rpoppler" Poppler PDF rendering library as provided by the functions PDF_info and PDF_text in package **Rpoppler**.

"ghostscript" Ghostscript using 'pdf_info.ps' and 'ps2ascii.ps'.

"Rcampdf" Perl CAM::PDF PDF manipulation library as provided by the functions pdf_info and pdf_text in package **Rcampdf**, available from the repository at http://datacube.wu.ac.at.

"custom" custom user-provided extraction engine.

Control parameters for engine "xpdf" are as follows.

info a character vector specifying options passed over to the pdfinfo executable.

text a character vector specifying options passed over to the pdftotext executable.

Control parameters for engine "custom" are as follows.

info a function extracting meta data from a PDF. The function must accept a file path as first argument and must return a named list with the components Author (as character string), CreationDate (of class POSIXlt), Subject (as character string), Title (as character string), and Creator (as character string).

text a function extracting content from a PDF. The function must accept a file path as first argument and must return a character vector.

Value

A function with the signature elem, language, id:

elem a list with the named component uri which must hold a valid file name.

language a string giving the text's language.

id a unique identification string for the returned text document.

The function returns a PlainTextDocument representing the text and meta data extracted from elem$uri.

Author(s)

Ingo Feinerer

See Also

getReaders to list available reader functions.

Examples

```
uri <- system.file(file.path("doc", "tm.pdf"), package = "tm")
if(all(file.exists(Sys.which(c("pdfinfo", "pdftotext"))))) {
    pdf <- readPDF(control = list(text = "-layout"))(elem = list(uri = uri),
                                                     language = "en",
                                                     id = "id1")
    pdf[1:13]
}
Corpus(URISource(uri),
       readerControl = list(reader = readPDF(engine = "ghostscript")))
```

---

readPlain *Read In a Text Document*

Description

Read in a text document without knowledge about its internal structure and possible available metadata.

Usage

```
readPlain(elem, language, id)
```

Arguments

elem	a list with the named component content which must hold the document to be read in.
language	a string giving the text's language.
id	a unique identification string for the returned text document.

Value

A PlainTextDocument representing elem$content.

Author(s)

Ingo Feinerer

See Also

getReaders to list available reader functions.

Examples

```
docs <- c("This is a text.", "This another one.")
vs <- VectorSource(docs)
elem <- getElem(stepNext(vs))
(result <- readPlain(elem, "en", "id1"))
meta(result)
```

---

readRCV1 *Read In a Reuters Corpus Volume 1 Document*

Description

Read in a Reuters Corpus Volume 1 XML document.

Usage

```
readRCV1(elem, language, id)
readRCV1asPlain(elem, language, id)
```

Arguments

elem	a list with the named component content which must hold the document to be read in.
language	a string giving the text's language.
id	a unique identification string for the returned text document.

Value

An RCV1Document for readRCV1, or a PlainTextDocument for readRCV1asPlain.

Author(s)

Ingo Feinerer

References

Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, **5**, 361–397. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf

See Also

getReaders to list available reader functions.

Examples

```
f <- system.file("texts", "rcv1_2330.xml", package = "tm")
rcv1 <- readRCV1(elem = list(content = readLines(f)),
                 language = "en", id = "id1")
meta(rcv1)
```

---

readReut21578XML *Read In a Reuters-21578 XML Document*

Description

Read in a Reuters-21578 XML document.

Usage

```
readReut21578XML(elem, language, id)
readReut21578XMLasPlain(elem, language, id)
```

Arguments

elem	a list with the named component content which must hold the document to be read in.
language	a string giving the text's language.
id	a unique identification string for the returned text document.

Value

A Reuters21578Document for readReut21578XML, or a PlainTextDocument for readReut21578XMLasPlain.

Author(s)

Ingo Feinerer

References

Lewis, David (1997) *Reuters-21578 Text Categorization Collection Distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Luz, Saturnino XML-*encoded version of Reuters-21578*. http://modnlp.berlios.de/reuters21578.html

See Also

`getReaders` to list available reader functions.

---

readTabular *Read In a Text Document*

Description

Return a function which reads in a text document from a tabular data structure (like a data frame or a list matrix) with knowledge about its internal structure and possible available metadata as specified by a so-called mapping.

Usage

```
readTabular(mapping)
```

Arguments

`mapping`	A named `list` of `characters`. The constructed reader will map each character entry to the content or meta datum of the text document as specified by the named list entry. Valid names include `Content` to access the document's content, any valid attribute name, and characters which are mapped to `LocalMetaData` entries.

Details

Formally this function is a function generator, i.e., it returns a function (which reads in a text document) with a well-defined signature, but can access passed over arguments (e.g., the mapping) via lexical scoping.

Value

A `function` with the signature `elem, language, id`:

`elem` a list with the named component `content` which must hold the document to be read in.

`language` a string giving the text's language.

`id` a unique identification string for the returned text document.

The function returns a `PlainTextDocument` representing the text and meta data extracted from `elem$content`.

Author(s)

Ingo Feinerer

See Also

Vignette 'Extensions: How to Handle Custom File Formats'.

`getReaders` to list available reader functions.

Examples

```
df <- data.frame(contents = c("content 1", "content 2", "content 3"),
                 title    = c("title 1"  , "title 2"  , "title 3"  ),
                 authors  = c("author 1" , "author 2" , "author 3" ),
                 topics   = c("topic 1"  , "topic 2"  , "topic 3"  ),
                 stringsAsFactors = FALSE)
m <- list(Content = "contents", Heading = "title",
          Author = "authors", Topic = "topics")
myReader <- readTabular(mapping = m)
ds <- DataframeSource(df)
elem <- getElem(stepNext(ds))
(result <- myReader(elem, language = "en", id = "id1"))
meta(result)
```

---

readXML *Read In an XML Document*

Description

Return a function which reads in an XML document. The structure of the XML document can be described with a specification.

Usage

```
readXML(spec, doc)
```

Arguments

spec	A named list of lists each containing two components. The constructed reader will map each list entry to an attribute or meta datum corresponding to the named list entry. Valid names include `Content` to access the document's content, any valid attribute name, and characters which are mapped to `LocalMetaData` entries. Each list entry must consist of two components: the first must be a string describing the type of the second argument, and the second is the specification entry. Valid combinations are: `type = "node", spec = "XPathExpression"` The XPath expression `spec` extracts information from an XML node. `type = "attribute", spec = "XPathExpression"` The XPath expression `spec` extracts information from an attribute of an XML node. `type = "function", spec = function(tree) ...` The function `spec` is called, passing over a tree representation (as delivered by `xmlInternalTreeParse` from package **XML**) of the read in XML document as first argument. `type = "unevaluated", spec = "String"` The character vector `spec` is returned without modification.
doc	An (empty) document of some subclass of `TextDocument`

Details

Formally this function is a function generator, i.e., it returns a function (which reads in a text document) with a well-defined signature, but can access passed over arguments (e.g., the specification) via lexical scoping.

Value

A function with the signature `elem, language, id`:

`elem` a list with the named component `content` which must hold the document to be read in.

`language` a string giving the text's language.

`id` a unique identification string for the returned text document.

The function returns `doc` augmented by the parsed information as described by `spec` out of the XML file in `elem$content`.

Author(s)

Ingo Feinerer

See Also

Vignette 'Extensions: How to Handle Custom File Formats', `XMLSource`.

`getReaders` to list available reader functions.

Examples

```
readGmane <-
readXML(spec = list(Author = list("node", "/item/creator"),
                    Content = list("node", "/item/description"),
                    DateTimeStamp = list("function", function(node)
                    strptime(sapply(XML::getNodeSet(node, "/item/date"), XML::xmlValue),
                             format = "%Y-%m-%dT%H:%M:%S",
                             tz = "GMT")),
                    Description = list("unevaluated", ""),
                    Heading = list("node", "/item/title"),
                    ID = list("node", "/item/link"),
                    Origin = list("unevaluated", "Gmane Mailing List Archive")),
                    doc = PlainTextDocument())
```

---

removeNumbers *Remove Numbers from a Text Document*

Description

Stip any numbers from a text document.

Usage

```
## S3 method for class 'PlainTextDocument'
removeNumbers(x)
```

Arguments

x	A text document.

Value

The text document with any numbers in it removed.

See Also

getTransformations to list available transformation (mapping) functions.

Examples

```
data("crude")
crude[[1]]
removeNumbers(crude[[1]])
```

removePunctuation *Remove Punctuation Marks from a Text Document*

Description

Remove punctuation marks from a text document.

Usage

```
## S3 method for class 'PlainTextDocument'
removePunctuation(x, preserve_intra_word_dashes = FALSE)
```

Arguments

x	A text document.
preserve_intra_word_dashes	A logical specifying whether intra-word dashes should be kept.

Value

The text document x with any punctuation marks in it removed (besides intra-word dashes if preserve_intra_word_dashes is set).

See Also

getTransformations to list available transformation (mapping) functions.

regex shows the class [:punct:] of punctuation characters.

Examples

```
data("crude")
crude[[14]]
removePunctuation(crude[[14]])
removePunctuation(crude[[14]], preserve_intra_word_dashes = TRUE)
```

removeSparseTerms *Remove Sparse Terms from a Term-Document Matrix*

Description

Remove sparse terms from a document-term or term-document matrix.

Usage

```
removeSparseTerms(x, sparse)
```

Arguments

x	A DocumentTermMatrix or a TermDocumentMatrix.
sparse	A numeric for the maximal allowed sparsity in the range from bigger zero to smaller one.

Value

A term-document matrix where those terms from x are removed which have at least a sparse percentage of empty (i.e., terms occurring 0 times in a document) elements. I.e., the resulting matrix contains only terms with a sparse factor of less than sparse.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude)
removeSparseTerms(tdm, 0.2)
```

---

removeWords *Remove Words from a Text Document*

Description

Remove a set of words from a text document.

Usage

```
## S3 method for class 'PlainTextDocument'
removeWords(x, words)
```

Arguments

x	A text document.
words	A character vector listing the words to be removed.

Value

The text document with the specified words in it removed.

See Also

getTransformations to list available transformation (mapping) functions.

Examples

```
data("crude")
crude[[1]]
removeWords(crude[[1]], stopwords("english"))
```

`Reuters21578Document` *Reuters-21578 Text Document*

Description

Construct an object representing a Reuters-21578 XML text document with meta information.

Usage

```
Reuters21578Document(x, author = character(0),
                     datetimestamp = as.POSIXlt(Sys.time(), tz = "GMT"),
                     description = character(0), heading = character(0),
                     id = character(0), origin = character(0),
                     language = character(0), localmetadata = list())
```

Arguments

`x`	Object of class `list` containing the content.
`author`	Object of class `character` containing the author names.
`datetimestamp`	Object of class `POSIXlt` containing the date and time when the document was written.
`description`	Object of class `character` containing additional text information.
`heading`	Object of class `character` containing the title or a short heading.
`id`	Object of class `character` containing an identifier.
`origin`	Object of class `character` containing information on the source and origin of the text.
`language`	Object of class `character` containing the language of the text (preferably as IETF language tags).
`localmetadata`	Object of class `list` containing local meta data in form of tag-value pairs.

Author(s)

Ingo Feinerer

References

Lewis, David (1997) *Reuters-21578 Text Categorization Collection Distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Luz, Saturnino XML-*encoded version of Reuters-21578*. http://modnlp.berlios.de/reuters21578.html

See Also

`PlainTextDocument` and `RCV1Document`

ReutersSource *Reuters-21578 XML Source*

Description

Construct a source for an input containing several Reuters-21578 XML documents.

Usage

```
ReutersSource(x, encoding = "unknown")
```

Arguments

x	Either a character identifying the file or a connection.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Value

An object of class XMLSource which extends the class Source representing a Reuters-21578 XML document.

Author(s)

Ingo Feinerer

References

Lewis, David (1997) *Reuters-21578 Text Categorization Collection Distribution 1.0*. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

Luz, Saturnino XML-*encoded version of Reuters-21578*. http://modnlp.berlios.de/reuters21578.html

See Also

getSources to list available sources. Encoding on encodings in R.

Examples

```
reuters21578 <- system.file("texts", "reuters-21578.xml", package = "tm")
rs <- ReutersSource(reuters21578)
inspect(Corpus(rs)[1:2])
```

sFilter *Statement Filter*

Description

Filter meta data by user-defined statements.

Usage

```
sFilter(x, s)
```

Arguments

x	A Corpus.
s	A statement of format "tag1 == 'expr1' & tag2 == 'expr2' & ...".

Details

The statement s models a simple query language. It consists of an expression as passed over to a data frame for subsetting. Tags in s represent meta data variables. Variables only available at document level are shifted up to the data frame if necessary. Note that the meta data tags for the slots Author, DateTimeStamp, Description, ID, Origin and Heading are author, datetimestamp, description, id, origin and heading, respectively, to avoid name conflicts.

Value

A logical vector to represent the subset of the DMetaData (extended for shifted up variables) data frame as specified by the statement.

Author(s)

Ingo Feinerer

Examples

```
data("crude")
sFilter(crude, "id == '127' & heading == 'DIAMOND SHAMROCK (DIA) CUTS CRUDE PRICES'")
sFilter(crude, "Places == 'usa'")
```

Source *Create and Access Sources*

Description

Create and access sources which abstract input locations, like a directory, a connection, or simply an R vector.

Usage

```
Source(defaultReader = readPlain, encoding = "unknown", length = NA_integer_,
       names = NA_character_, position = 0, vectorized = TRUE, class)
is.Source(x)
## S3 method for class 'Source'
eoi(x)
## S3 method for class 'DataframeSource'
getElem(x)
## S3 method for class 'DirSource'
getElem(x)
## S3 method for class 'URISource'
getElem(x)
## S3 method for class 'VectorSource'
getElem(x)
## S3 method for class 'XMLSource'
getElem(x)
## S3 method for class 'DataframeSource'
pGetElem(x)
## S3 method for class 'DirSource'
pGetElem(x)
## S3 method for class 'VectorSource'
pGetElem(x)
## S3 method for class 'Source'
stepNext(x)
```

Arguments

x	a source.
defaultReader	a reader function (generator).
encoding	a character specifying the encoding of the elements delivered by the source.
length	an integer denoting the number of elements delivered by the source. If the length is unknown in advance it must be set to NA_integer_.
names	a character vector giving element names.
position	a numeric indicating the position in the source.
vectorized	a logical indicating the ability for parallel element access.
class	a character vector for extending the class attribute.

Details

The function `Source` is a constructor and should be used when creating custom sources. Internally a source is represented as a `list` with the class attribute `Source`. Custom sources may extend the internal list structure with additional named components.

The function `is.Source` returns `TRUE` for a valid source and `FALSE` otherwise.

Each source must provide implementations for the three interface functions `eoi`, `getElem`, and `stepNext`. If `vectorized` is set `pGetElem` must be implemented as well. The function `eoi` returns `TRUE` if the end of input of the source is reached. `getElem` fetches the element at the current position, whereas `pGetElem` retrieves all elements in parallel at once. Retrieved elements must be encapsulated in a list with the named components `content` holding the document and `uri` pointing to the origin of the document (e.g., a file path or a connection; `NA` if not applicable or unavailable). `stepNext` increases the position in the source to the next element.

Author(s)

Ingo Feinerer

See Also

`getSources` to list available sources; `readPlain`; `Encoding`.

---

stemCompletion *Complete Stems*

Description

Heuristically complete stemmed words.

Usage

```
## S3 method for class 'character'
stemCompletion(x, dictionary,
               type = c("prevalent", "first", "longest",
                        "none", "random", "shortest"))
## S3 method for class 'PlainTextDocument'
stemCompletion(x, dictionary,
               type = c("prevalent", "first", "longest",
                        "none", "random", "shortest"))
```

Arguments

`x`	A `PlainTextDocument` or `character` vector of stems to be completed.
`dictionary`	A `Corpus` or `character` vector to be searched for possible completions.
`type`	A `character` naming the heuristics to be used:
	`prevalent` Default. Takes the most frequent match as completion.

`first` Takes the first found completion.
`longest` Takes the longest completion in terms of characters.
`none` Is the identity.
`random` Takes some completion.
`shortest` Takes the shortest completion in terms of characters.

Value

A plain text document or character vector with completed words.

Author(s)

Ingo Feinerer

References

Ingo Feinerer (2010). Analysis and Algorithms for Stemming Inversion. *Information Retrieval Technology — 6th Asia Information Retrieval Societies Conference, AIRS 2010, Taipei, Taiwan, December 1–3, 2010. Proceedings*, volume 6458 of *Lecture Notes in Computer Science*, pages 290–299. Springer-Verlag, December 2010.

Examples

```
data("crude")
stemCompletion(c("compan", "entit", "suppl"), crude)
(s <- stemDocument(crude[[1]]))
stemCompletion(s, crude)
```

---

stemDocument *Stem Words*

Description

Stem words in a text document using Porter's stemming algorithm.

Usage

```
## S3 method for class 'PlainTextDocument'
stemDocument(x, language = Language(x))
```

Arguments

`x`	A text document.
`language`	A character setting the language to be used for stemming.

Details

The argument `language` is passed over to `wordStem` as the name of the Snowball stemmer.

Examples

```
data("crude")
crude[[1]]
stemDocument(crude[[1]])
```

`stopwords` *Stopwords*

Description

Return various kinds of stopwords with support for different languages.

Usage

```
stopwords(kind = "en")
```

Arguments

`kind`	A `character` identifying the desired stopword list.

Details

Available stopword lists are:

`SMART` English stopwords from the SMART information retrieval system (obtained from http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop) (which coincides with the stopword list used by the MC toolkit (http://www.cs.utexas.edu/users/dml/software/mc/)),

`catalan` Catalan stopwords (obtained from http://latel.upf.edu/morgana/altres/pub/ca_stop.htm), and

and a set of stopword lists from the Snowball stemmer project in different languages (obtained from http://svn.tartarus.org/snowball/trunk/website/algorithms/*/stop.txt). Supported languages are `danish`, `dutch`, `english`, `finnish`, `french`, `german`, `hungarian`, `italian`, `norwegian`, `portuguese`, `russian`, `spanish`, and `swedish`. Language names are case sensitive. Alternatively, their IETF language tags may be used.

Value

A character vector containing the requested stopwords. An error is raised if no stopwords are available for the requested `kind`.

Examples

```
stopwords("en")
stopwords("SMART")
stopwords("german")
```

stripWhitespace *Strip Whitespace from a Text Document*

Description

Strip extra whitespace from a text document. Multiple white space characters are collapsed to a single blank.

Usage

```
## S3 method for class 'PlainTextDocument'
stripWhitespace(x)
```

Arguments

x	A text document.

Value

The text document with multiple white space characters collapse to a single blank.

See Also

getTransformations to list available transformation (mapping) functions.

Examples

```
data("crude")
crude[[1]]
stripWhitespace(crude[[1]])
```

TermDocumentMatrix *Term-Document Matrix*

Description

Constructs or coerces to a term-document matrix or a document-term matrix.

Usage

```
TermDocumentMatrix(x, control = list())
DocumentTermMatrix(x, control = list())
as.TermDocumentMatrix(x, ...)
as.DocumentTermMatrix(x, ...)
```

Arguments

x	a corpus for the constructors and either a term-document matrix or a document-term matrix or a simple triplet matrix (package **slam**) or a term frequency vector for the coercing functions.
control	a named list of control options. There are local options which are evaluated for each document and global options which are evaluated once for the constructed matrix. Available local options are documented in termFreq and are internally delegated to a termFreq call. Available global options are: bounds A list with a tag global whose value must be an integer vector of length 2. Terms that appear in less documents than the lower bound bounds$global[1] or in more documents than the upper bound bounds$global[2] are discarded. Defaults to list(global = c(1,Inf)) (i.e., every term will be used). weighting A weighting function capable of handling a TermDocumentMatrix. It defaults to weightTf for term frequency weighting. Available weighting functions shipped with the **tm** package are weightTf, weightTfIdf, weightBin, and weightSMART.
...	the additional argument weighting (typically a WeightFunction) is allowed when coercing a simple triplet matrix to a term-document or document-term matrix.

Value

An object of class TermDocumentMatrix or class DocumentTermMatrix (both inheriting from a simple triplet matrix in package **slam**) containing a sparse term-document matrix or document-term matrix. The attribute Weighting contains the weighting applied to the matrix.

Author(s)

Ingo Feinerer

See Also

termFreq for available local control options.

Examples

```
data("crude")
tdm <- TermDocumentMatrix(crude,
                          control = list(removePunctuation = TRUE,
                                         stopwords = TRUE))
dtm <- DocumentTermMatrix(crude,
                          control = list(weighting =
                                         function(x)
                                         weightTfIdf(x, normalize =
                                                     FALSE),
                                         stopwords = TRUE))
inspect(tdm[155:160,1:5])
inspect(tdm[c("price", "texas"),c("127","144","191","194")])
```

```
inspect(dtm[1:5,155:160])
```

---

termFreq *Term Frequency Vector*

Description

Generate a term frequency vector from a text document.

Usage

```
termFreq(doc, control = list())
```

Arguments

doc	An object inheriting from TextDocument.
control	A list of control options which override default settings.

First, following two options are processed.

`tolower` Either a logical value indicating whether characters should be translated to lower case or a custom function converting characters to lower case. Defaults to `tolower`.

`tokenize` A function tokenizing documents into single tokens or a string matching one of the predefined tokenization functions:

`scan` for `scan_tokenizer`, or

`MC` for `MC_tokenizer`.

Defaults to `scan_tokenizer`.

Next, a set of options which are sensitive to the order of occurrence in the `control` list. Options are processed in the same order as specified. User-specified options have precedence over the default ordering so that first all user-specified options and then all remaining options (with the default settings and in the order as listed below) are processed.

`removePunctuation` A logical value indicating whether punctuation characters should be removed from doc, a custom function which performs punctuation removal, or a list of arguments for `removePunctuation`. Defaults to `FALSE`.

`removeNumbers` A logical value indicating whether numbers should be removed from doc or a custom function for number removal. Defaults to `FALSE`.

`stopwords` Either a Boolean value indicating stopword removal using default language specific stopword lists shipped with this package, a character vector holding custom stopwords, or a custom function for stopword removal. Defaults to `FALSE`.

`stemming` Either a Boolean value indicating whether tokens should be stemmed or a custom stemming function. Defaults to `FALSE`.

Finally, following options are processed in the given order.

`dictionary` A character vector to be tabulated against. No other terms will be listed in the result. Defaults to `NULL` which means that all terms in `doc` are listed.

`bounds` A list with a tag `local` whose value must be an integer vector of length 2. Terms that appear less often in `doc` than the lower bound `bounds$local[1]` or more often than the upper bound `bounds$local[2]` are discarded. Defaults to `list(local = c(1,Inf))` (i.e., every token will be used).

`wordLengths` An integer vector of length 2. Words shorter than the minimum word length `wordLengths[1]` or longer than the maximum word length `wordLengths[2]` are discarded. Defaults to `c(3, Inf)`, i.e., a minimum word length of 3 characters.

Value

A named integer vector of class `term_frequency` with term frequencies as values and tokens as names.

See Also

`getTokenizers`

Examples

```
data("crude")
termFreq(crude[[14]])
strsplit_space_tokenizer <- function(x) unlist(strsplit(x, "[[:space:]]+"))
ctrl <- list(tokenize = strsplit_space_tokenizer,
             removePunctuation = list(preserve_intra_word_dashes = TRUE),
             stopwords = c("reuter", "that"),
             stemming = TRUE,
             wordLengths = c(4, Inf))
termFreq(crude[[14]], control = ctrl)
```

TextDocument *Access and Modify Text Documents*

Description

Access the meta data of text documents, and access and modify the content of text documents.

Details

The class `TextDocument` provides an abstraction over the concept of text documents and attached meta data which is stored in following attributes:

`Author` Object of class `character` containing the author names.

`DateTimeStamp` Object of class `POSIXlt` containing the date and time when the document was written.

`Description` Object of class `character` containing additional text information.

`ID` Object of class `character` containing an identifier.

`Origin` Object of class `character` containing information on the source and origin of the text.

`Heading` Object of class `character` containing the title or a short heading.

`Language` Object of class `character` containing the language of the text.

`LocalMetaData` Object of class `list` containing additional meta data in form of tag-value pairs which is local to each individual text document.

Author(s)

Ingo Feinerer

See Also

`meta` and `DublinCore` which provide a unified interface for meta data management.

`TextRepository` *Text Repository*

Description

Construct a text repository for corpora. A repository is designed to keep track of multiple corpora, either different ones, or corpora with the same underlying texts but at different preprocessing stages.

Usage

```
TextRepository(x,
               meta = list(created =
                           as.POSIXlt(Sys.time(), tz = "GMT")))
```

Arguments

`x`	A corpus.
`meta`	An initial list of tag-value pairs for the repository meta data.

Value

An object of class `TextRepository` which extends the class `list` containing corpora. Meta data annotations are stored in the attribute `RepoMetaData` in form of tag-value pairs (i.e., a named list).

Author(s)

Ingo Feinerer

Examples

```
data("crude")
repo <- TextRepository(crude)
summary(repo)
RepoMetaData(repo)
```

`tm_combine` *Combine Corpora, Documents, Term-Document Matrices, and Term Frequency Vectors*

Description

Combine several corpora into a single one, combine multiple documents into a corpus, combine multiple term-document matrices into a single one, or combine multiple term frequency vectors into a single term-document matrix.

Usage

```
## S3 method for class 'Corpus'
c(..., recursive = FALSE)
## S3 method for class 'TextDocument'
c(..., recursive = FALSE)
## S3 method for class 'TermDocumentMatrix'
c(..., recursive = FALSE)
## S3 method for class 'term_frequency'
c(..., recursive = FALSE)
```

Arguments

`...`	Corpora, text documents, term-document matrices, or term frequency vectors.
`recursive`	Logical. If `recursive = TRUE` existing corpus meta data is also merged, otherwise discarded.

Details

If `recursive = TRUE`, meta data from input objects (corpora or documents) is preserved during concatenation and intelligently merged into the newly created corpus. Although we use a sophisticated merging strategy (by using a binary tree for corpus specific meta data and by joining document level specific meta data in data frames) you should check the newly created meta data for consistency when merging corpora with (partly) identical meta data. However, in most cases the meta data merging strategy will produce validly combined and arranged meta data structures.

See Also

`Corpus`, `TextDocument`, `TermDocumentMatrix`, and `termFreq`.

Examples

```
data("acq")
data("crude")
summary(c(acq, crude))
summary(c(acq[[30]], crude[[10]]))
c(TermDocumentMatrix(acq), TermDocumentMatrix(crude))
```

---

tm_filter *Filter and Index Functions on Corpora*

Description

Interface to apply filter and index functions to corpora.

Usage

```
## S3 method for class 'Corpus'
tm_filter(x, ..., FUN, doclevel = TRUE, useMeta = FALSE)
## S3 method for class 'Corpus'
tm_index(x, ..., FUN, doclevel = TRUE, useMeta = FALSE)
```

Arguments

x	A corpus.
...	Arguments to FUN.
FUN	A filter function returning a logical value.
doclevel	Logical. If the document level flag is set FUN is applied to each element of x, otherwise FUN is applied to x itself. If FUN has an attribute doclevel its value will be automatically used.
useMeta	Logical. Should DMetaData be passed over to FUN as argument?

Value

tm_filter returns a corpus containing documents where FUN matches, whereas tm_index only returns the corresponding indices.

See Also

sFilter for a filter using a simple statement query language.

Examples

```
data("crude")
# Full-text search
tm_filter(crude, FUN = function(x) any(grep("co[m]?pany", x)))
```

tm_map *Transformations on Corpora*

Description

Interface to apply transformation functions (also denoted as mappings) to corpora.

Usage

```
## S3 method for class 'PCorpus'
tm_map(x, FUN, ..., useMeta = FALSE, lazy = FALSE)
## S3 method for class 'VCorpus'
tm_map(x, FUN, ..., useMeta = FALSE, lazy = FALSE)
```

Arguments

x	A corpus.
FUN	A transformation function returning a text document.
...	Arguments to FUN.
useMeta	Logical. Should DMetaData be passed over to FUN as argument?
lazy	Logical. Lazy mappings are mappings which are delayed until the documents' content is accessed. Lazy mapping is useful when working with large corpora but only few documents will be accessed, as it avoids the computationally expensive application of the mapping to all elements in the corpus.

Value

A corpus with FUN applied to each document in x. In case of lazy mappings only annotations are stored which are evaluated upon access of individual documents which trigger the execution of the corresponding transformation function.

Note

Please be aware that lazy transformations are an experimental feature and change R's standard evaluation semantics.

See Also

getTransformations for available transformations, and materialize for manually triggering the materialization of documents with pending lazy transformations.

Examples

```
data("crude")
tm_map(crude, stemDocument)
## Generate a custom transformation function which takes the heading
## as new content
headings <- function(x)
    PlainTextDocument(Heading(x), id = ID(x), language = Language(x))
inspect(tm_map(crude, headings))
```

`tm_reduce` *Combine Transformations*

Description

Fold multiple transformations (mappings) into a single one.

Usage

```
tm_reduce(x, tmFuns, ...)
```

Arguments

`x`	A corpus.
`tmFuns`	A `list` of **tm** transformations.
`...`	Arguments to the individual transformations.

Value

A single **tm** transformation function obtained by folding `tmFuns` from right to left (via `Reduce(..., right = TRUE)`).

Author(s)

Ingo Feinerer

See Also

`Reduce` for R's internal folding/accumulation mechanism, and `getTransformations` to list available transformation (mapping) functions.

Examples

```
data(crude)
crude[[1]]
skipWords <- function(x) removeWords(x, c("it", "the"))
funs <- list(stripWhitespace, skipWords, removePunctuation, tolower)
tm_map(crude, FUN = tm_reduce, tmFuns = funs)[[1]]
```

`tm_term_score` *Compute Score for Matching Terms*

Description

Compute a score based on the number of matching terms.

Usage

```
## S3 method for class 'DocumentTermMatrix'
tm_term_score(x, terms, FUN = slam::row_sums)
## S3 method for class 'PlainTextDocument'
tm_term_score(x, terms, FUN = function(x) sum(x, na.rm = TRUE))
## S3 method for class 'term_frequency'
tm_term_score(x, terms, FUN = function(x) sum(x, na.rm = TRUE))
## S3 method for class 'TermDocumentMatrix'
tm_term_score(x, terms, FUN = slam::col_sums)
```

Arguments

`x`	Either a `PlainTextDocument`, a term frequency as returned by `termFreq`, or a `TermDocumentMatrix`.
`terms`	A character vector of terms to be matched.
`FUN`	A function computing a score from the number of terms matching in `x`.

Value

A score as computed by `FUN` from the number of matching `terms` in `x`.

Examples

```
data("acq")
tm_term_score(acq[[1]], c("company", "change"))
## Not run: ## Test for positive and negative sentiments
## install.packages("tm.lexicon.GeneralInquirer", repos="http://datacube.wu.ac.at", type="source")
require("tm.lexicon.GeneralInquirer")
sapply(acq[1:10], tm_term_score, terms_in_General_Inquirer_categories("Positiv"))
sapply(acq[1:10], tm_term_score, terms_in_General_Inquirer_categories("Negativ"))
tm_term_score(TermDocumentMatrix(acq[1:10],
                                 control = list(removePunctuation = TRUE)),
              terms_in_General_Inquirer_categories("Positiv"))
## End(Not run)
```

tokenizer *Tokenizers*

Description

Tokenize a document or character vector.

Usage

```
MC_tokenizer(x)
scan_tokenizer(x)
```

Arguments

x	A character vector.

Details

The quality and correctness of a tokenization algorithm highly depends on the context and application scenario. Relevant factors are the language of the underlying text and the notions of whitespace (which can vary with the used encoding and the language) and punctuation marks. Consequently, for superior results you probably need a custom tokenization function.

scan_tokenizer Relies on `scan(..., what = "character")`.

MC_tokenizer Implements the functionality of the tokenizer in the MC toolkit (http://www.cs.utexas.edu/users/dml/software/mc/).

Value

A `character` vector consisting of tokens obtained by tokenization of `x`.

Author(s)

Ingo Feinerer

See Also

getTokenizers

Examples

```
data("crude")
MC_tokenizer(crude[[1]])
scan_tokenizer(crude[[1]])
strsplit_space_tokenizer <- function(x) unlist(strsplit(x, "[[:space:]]+"))
strsplit_space_tokenizer(crude[[1]])
```

URISource *Uniform Resource Identifier Source*

Description

Constructs a source which represents documents located by a uniform resource identifier.

Usage

```
URISource(x, encoding = "unknown")
```

Arguments

x	A vector of Uniform Resource Identifier, i.e., either a character identifying the file or a connection.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Value

An object of class URISource which extends the class Source representing documents located by a URI.

Author(s)

Ingo Feinerer

See Also

DirSource for accessing a directory, and getSources to list available sources. Encoding on encodings in R.

Examples

```
loremipsum <- system.file("texts", "loremipsum.txt", package = "tm")
ovid <- system.file("texts", "txt", "ovid_1.txt", package = "tm")
us <- URISource(c(loremipsum, ovid))
inspect(Corpus(us))
```

VCorpus *Volatile Corpus*

Description

Data structures and operators for volatile corpora.

Usage

```
Corpus(x, readerControl = list(reader = x$DefaultReader, language = "en"))
VCorpus(x, readerControl = list(reader = x$DefaultReader, language = "en"))
## S3 method for class 'VCorpus'
DMetaData(x)
## S3 method for class 'Corpus'
CMetaData(x)
```

Arguments

x	A Source object for Corpus and VCorpus, and a corpus for the other functions.
readerControl	A list with the named components reader representing a reading function capable of handling the file format found in x, and language giving the text's language (preferably as IETF language tags). The default language is assumed to be English ("en"). Use NA to avoid internal assumptions (e.g., when the language is unknown or is deliberately not set).

Details

Volatile means that the corpus is fully kept in memory and thus all changes only affect the corresponding R object. In contrast there is also a corpus implementation available providing a permanent semantics (see PCorpus).

The constructed corpus object inherits from a list and has two attributes containing meta information:

CMetaData Corpus Meta Data contains corpus specific meta data in form of tag-value pairs and information about children in form of a binary tree. This information is useful for reconstructing meta data after e.g. merging corpora.

DMetaData Document Meta Data of class data.frame contains document specific meta data for the corpus. This data frame typically encompasses clustering or classification results which basically are metadata for documents but form an own entity (e.g., with its name, the value range, etc.).

Value

An object of class VCorpus which extends the classes Corpus and list containing a collection of text documents.

Author(s)

Ingo Feinerer

Examples

```
reut21578 <- system.file("texts", "crude", package = "tm")
(r <- Corpus(DirSource(reut21578),
             readerControl = list(reader = readReut21578XMLasPlain)))
```

VectorSource *Vector Source*

Description

Constructs a source for a vector as input.

Usage

```
VectorSource(x, encoding = "unknown")
```

Arguments

x	A vector.
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Value

An object of class `VectorSource` which extends the class `Source` representing a vector where each entry is interpreted as a document.

Author(s)

Ingo Feinerer

See Also

`getSources` to list available sources. `Encoding` on encodings in R.

Examples

```
docs <- c("This is a text.", "This another one.")
(vs <- VectorSource(docs))
inspect(Corpus(vs))
```

weightBin *Weight Binary*

Description

Binary weight a term-document matrix.

Usage

```
weightBin(m)
```

Arguments

m	A TermDocumentMatrix in term frequency format.

Details

Formally this function is of class WeightingFunction with the additional attributes Name and Acronym.

Value

The weighted matrix.

Author(s)

Ingo Feinerer

WeightFunction *Weighting Function*

Description

Construct a weighting function for term-document matrices.

Usage

```
WeightFunction(x, name, acronym)
```

Arguments

x	A function which takes a TermDocumentMatrix with term frequencies as input, weights the elements, and returns the weighted matrix.
name	A character naming the weighting function.
acronym	A character giving an acronym for the name of the weighting function.

Value

An object of class `WeightFunction` which extends the class `function` representing a weighting function.

Author(s)

Ingo Feinerer

Examples

```
weightCutBin <- WeightFunction(function(m, cutoff) m > cutoff,
                               "binary with cutoff", "bincut")
```

`weightSMART` *SMART Weightings*

Description

Weight a term-document matrix according to a combination of weights specified in SMART notation.

Usage

```
weightSMART(m, spec = "nnn", control = list())
```

Arguments

`m`	A `TermDocumentMatrix` in term frequency format.
`spec`	a character string consisting of three characters. The first letter specifies a term frequency schema, the second a document frequency schema, and the third a normalization schema. See **Details** for available built-in schemata.
`control`	a list of control parameters. See **Details**.

Details

Formally this function is of class `WeightingFunction` with the additional attributes `Name` and `Acronym`.

The first letter of `spec` specifies a weighting schema for term frequencies of `m`:

"n" (natural) $tf_{i,j}$ counts the number of occurrences $n_{i,j}$ of a term t_i in a document d_j. The input term-document matrix `m` is assumed to be in this standard term frequency format already.

"l" (logarithm) is defined as $1 + \log(tf_{i,j})$.

"a" (augmented) is defined as $0.5 + \frac{0.5 * tf_{i,j}}{\max_i(tf_{i,j})}$.

"b" (boolean) is defined as 1 if $tf_{i,j} > 0$ and 0 otherwise.

"L" (log average) is defined as $\frac{1+\log(tf_{i,j})}{1+\log(\text{ave}_{i \in j}(tf_{i,j}))}$.

The second letter of `spec` specifies a weighting schema of document frequencies for `m`:

"n" (no) is defined as 1.

"t" (idf) is defined as $\log \frac{N}{df_t}$ where df_t denotes how often term t occurs in all documents.

"p" (prob idf) is defined as $\max(0, \log(\frac{N - df_t}{df_t}))$.

The third letter of `spec` specifies a schema for normalization of `m`:

"n" (none) is defined as 1.

"c" (cosine) is defined as $\sqrt{\text{col\_sums}(m^2)}$.

"u" (pivoted unique) is defined as $slope * \sqrt{\text{col\_sums}(m^2)} + (1 - slope) * pivot$ where both `slope` and `pivot` must be set via named tags in the `control` list.

"b" (byte size) is defined as $\frac{1}{CharLength^\alpha}$. The parameter α must be set via the named tag `alpha` in the `control` list.

The final result is defined by multiplication of the chosen term frequency component with the chosen document frequency component with the chosen normalization component.

Value

The weighted matrix.

Author(s)

Ingo Feinerer

References

Christopher D. Manning and Prabhakar Raghavan and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press, ISBN 0521865719.

Examples

```
data("crude")
TermDocumentMatrix(crude,
                   control = list(removePunctuation = TRUE,
                                  stopwords = TRUE,
                                  weighting = function(x)
                                  weightSMART(x, spec = "ntc")))
```

weightTf *Weight by Term Frequency*

Description

Weight a term-document matrix by term frequency.

Usage

```
weightTf(m)
```

Arguments

m	A TermDocumentMatrix in term frequency format.

Details

Formally this function is of class WeightingFunction with the additional attributes Name and Acronym.

This function acts as the identity function since the input matrix is already in term frequency format.

Value

The weighted matrix.

Author(s)

Ingo Feinerer

weightTfIdf *Weight by Term Frequency - Inverse Document Frequency*

Description

Weight a term-document matrix by term frequency - inverse document frequency.

Usage

```
weightTfIdf(m, normalize = TRUE)
```

Arguments

m	A TermDocumentMatrix in term frequency format.
normalize	A Boolean value indicating whether the term frequencies should be normalized.

Details

Formally this function is of class `WeightingFunction` with the additional attributes `Name` and `Acronym`.

Term frequency $tf_{i,j}$ counts the number of occurrences $n_{i,j}$ of a term t_i in a document d_j. In the case of normalization, the term frequency $tf_{i,j}$ is divided by $\sum_k n_{k,j}$.

Inverse document frequency for a term t_i is defined as

$$idf_i = \log \frac{|D|}{|\{d \mid t_i \in d\}|}$$

where $|D|$ denotes the total number of documents and where $|\{d \mid t_i \in d\}|$ is the number of documents where the term t_i appears.

Term frequency - inverse document frequency is now defined as $tf_{i,j} \cdot idf_i$.

Value

The weighted matrix.

Author(s)

Ingo Feinerer

References

Gerard Salton and Christopher Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, **24**/5, 513–523.

`writeCorpus` *Write a Corpus to Disk*

Description

Write a plain text representation of a corpus to multiple files on disk corresponding to the individual documents in the corpus.

Usage

```
writeCorpus(x, path = ".", filenames = NULL)
```

Arguments

`x`	A corpus.
`path`	A character listing the directory to be written into.
`filenames`	Either `NULL` or a character vector. In case no filenames are provided, filenames are automatically generated by using the documents' ID strings in `x`.

Examples

```
data("crude")
## Not run: writeCorpus(crude, path = ".",
            filenames = paste(seq_along(crude), ".txt", sep = ""))
## End(Not run)
```

`XMLSource` *XML Source*

Description

Constructs a source for an XML file.

Usage

```
XMLSource(x, parser, reader, encoding = "unknown")
```

Arguments

`x`	a file name or a connection.
`parser`	a function accepting an XML tree (as delivered by `xmlTreeParse` in package **XML**) as input and returning a list of XML elements.
`reader`	a function capable of turning XML elements as returned by `parser` into a subclass of `TextDocument`.
`encoding`	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input.

Value

An object of class `XMLSource` which extends the class `Source` representing an XML file.

Author(s)

Ingo Feinerer

See Also

Vignette 'Extensions: How to Handle Custom File Formats', `readXML`; `Encoding`

Examples

```
## An implementation for readGmane is provided as an example in ?readXML
example(readXML)

## Construct a source for a Gmane mailing list RSS feed.
GmaneSource <-
function(x, encoding = "unknown")
  XMLSource(x, function(tree)
              XML::xmlChildren(XML::xmlRoot(tree))
                [names(XML::xmlChildren(XML::xmlRoot(tree))) == "item"],
            readGmane, encoding)

## Not run: gs <- GmaneSource("http://rss.gmane.org/gmane.comp.lang.r.general")
elem <- getElem(stepNext(gs))
(gmane <- readGmane(elem, language = "en", id = "id1"))
meta(gmane)
## End(Not run)
```

`Zipf_n_Heaps` *Explore Corpus Term Frequency Characteristics*

Description

Explore Zipf's law and Heaps' law, two empirical laws in linguistics describing commonly observed characteristics of term frequency distributions in corpora.

Usage

```
Zipf_plot(x, type = "l", ...)
Heaps_plot(x, type = "l", ...)
```

Arguments

`x`	a document-term matrix or term-document matrix with unweighted term frequencies.
`type`	a character string indicating the type of plot to be drawn, see `plot`.
`...`	further graphical parameters to be used for plotting.

Details

Zipf's law (e.g., http://en.wikipedia.org/wiki/Zipf%27s_law) states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table, or, more generally, that the pmf of the term frequencies is of the form $ck^{-\beta}$, where k is the rank of the term (taken from the most to the least frequent one). We can conveniently explore the degree to which the law holds by plotting the logarithm of the frequency against the logarithm of the rank, and inspecting the goodness of fit of a linear model.

Heaps' law (e.g., http://en.wikipedia.org/wiki/Heaps%27_law) states that the vocabulary size V (i.e., the number of different terms employed) grows polynomially with the text size T

(the total number of terms in the texts), so that $V = cT^{\beta}$. We can conveniently explore the degree to which the law holds by plotting $\log(V)$ against $\log(T)$, and inspecting the goodness of fit of a linear model.

Value

The coefficients of the fitted linear model. As a side effect, the corresponding plot is produced.

Examples

```
data("acq")
m <- DocumentTermMatrix(acq)
Zipf_plot(m)
Heaps_plot(m)
```

Index