

Package ‘tm.plugin.dc’

July 22, 2025

Version 0.2-10

Date 2020-11-29

Title Text Mining Distributed Corpus Plug-in

Description A plug-in for the text mining framework tm to support text mining in a distributed way. The package provides a convenient interface for handling distributed corpus objects based on distributed list objects.

License GPL (>= 2)

Depends DSL (>= 0.1-7), tm (>= 0.7)

Suggests XML

Imports NLP, slam (>= 0.1-22), utils

NeedsCompilation no

Author Ingo Feinerer [aut],
Stefan Theussl [aut, cre]

Maintainer Stefan Theussl <Stefan.Theussl@R-project.org>

Repository CRAN

Date/Publication 2020-11-29 14:00:03 UTC

Contents

DistributedCorpus	2
Revisions	3
TermDocumentMatrix.DCorpus	4
tm_map.DCorpus	5

Index	7
--------------	----------

DistributedCorpus *Distributed Corpus*

Description

Data structures and operators for distributed corpora.

Usage

```
DCorpus( x,
         readerControl = list(reader = reader(x),
                              language = "en"),
         storage = NULL, keep = TRUE, ... )
## S3 method for class 'DCorpus'
as.VCorpus(x)
as.DCorpus( x, storage = NULL, ... )
```

Arguments

<code>x</code>	for DCorpus, a Source object. At the moment only DirSource is supported. For <code>as.VCorpus()</code> and <code>as.DCorpus()</code> , an object to be coerced to a VCorpus/DCorpus. Currently coercion from/to classic tm corpora (VCorpus) is implemented.
<code>readerControl</code>	A list with the named components <code>reader</code> representing a reading function capable of handling the file format found in <code>x</code> , and <code>language</code> giving the text's language (preferably as IETF language tags, see language in package NLP).
<code>storage</code>	The storage subsystem to use with the DCorpus. Currently two types of storages are supported: local disk storage using the Local File System (LFS) and the Hadoop Distributed File System (HDFS). Default: 'LFS'.
<code>keep</code>	Should revisions be used when operating on the DCorpus? Default: TRUE
<code>...</code>	Optional arguments for the reader.

Details

When constructing a distributed corpus the input source is extracted via the supplied reader and stored on the given file system (argument `storage`). While the data set resides on the corresponding storage (e.g., HDFS), only a symbolic representation is held in R (a so-called [DList](#)) which allows to access the corpus via corresponding ([DList](#)) methods. Since the available memory for the distributed corpus is only restricted by available disk space in the given storage (and not main memory like in a standard **tm** corpus) by default we also store a set of so-called revisions, i.e., stages of the (processed) corpus. Revisions can be turned off later on using the [keepRevisions\(\)](#) replacement function.\

The constructed corpus object inherits from a **tm** [Corpus](#) and has several slots containing meta information:

`meta` Corpus Meta Data contains corpus specific meta data in form of tag-value pairs.

dmeta Document Meta Data of class `data.frame` contains document specific meta data for the corpus. This is mainly available to be compatible with standard **tm** corpus definitions but not yet actually used in the distributed scenario.

keep A logical indicating whether revisions representing stages e.g., in a preprocessing chain should be kept or not.

Value

An object inheriting from `DCorpus` and `Corpus`.

Author(s)

Ingo Feinerer and Stefan Theussl

See Also

[Corpus](#) for basic information on the corpus infrastructure employed by package **tm**.

Examples

```
## Similar to example in package 'tm'
reut21578 <- system.file("texts", "crude", package = "tm")
dc <- DistributedCorpus(DirSource(reut21578),
  readerControl = list(reader = readReut21578XMLasPlain) )
dc

## Coercion
data("crude")
as.DistributedCorpus(crude)
as.VCorpus(dc)
```

Revisions

Revisions of a Distributed Corpus

Description

Each modification of the documents in the corpus results in a new stage, i.e., *revision* of the corpus. To allow fast switching between multiple revisions all modifications may be kept on the file system. The function `setRevision()` allows to go back to any stage in the history of the corpus. The function `keepRevisions()` shows if revisions are turned on or off; the corresponding replacement function is used to set the desired behavior.

Usage

```
getRevisions( corpus )
removeRevision( corpus, revision )
setRevision( corpus, revision )
keepRevisions( corpus )
`keepRevisions<-`( corpus, value )
```

Arguments

corpus	A distributed corpus of class DCorpus.
revision	The revision which is to be set as active or removed.
value	A logical indicating whether revisions should be kept or not.

Value

Whereas `getRevisions()` returns a list of character strings naming all available revisions, `setRevision()` returns the distributed corpus with the given revision marked as active. The function `keepRevisions()` returns a logical indicating whether revisions are used or not.

Examples

```
## provide data on storage
data("crude")
dc <- as.DCorpus(crude)
## do some preprocessing
dc <- tm_map(dc, content_transformer(tolower))
## retrieve available revisions
revs <- getRevisions(dc)
revs
## go back to original revision
setRevision(dc, revs[2])
keepRevisions(dc)
keepRevisions(dc) <- FALSE
```

TermDocumentMatrix.DCorpus

Term-Document Matrix from Distributed Corpora

Description

Constructs a term-document matrix given a distributed corpus.

Usage

```
## S3 method for class 'DCorpus'
TermDocumentMatrix(x, control = list())
```

Arguments

x	A distributed corpus.
control	A named list of control options. The component weighting must be a weighting function capable of handling a TermDocumentMatrix. It defaults to <code>weightTf</code> for term frequency weighting. All other options are delegated internally to a termFreq call.

Value

An object of class `TermDocumentMatrix` containing a sparse term-document matrix. The attribute `Weighting` contains the weighting applied to the matrix.

See Also

The documentation of [termFreq](#) gives an extensive list of possible options.

[TermDocumentMatrix](#)

Examples

```
data("crude")
tdm <- TermDocumentMatrix(as.DCorpus(crude),
                           list(stopwords = TRUE, weighting = weightTfIdf))
inspect(tdm[149:152,1:5])
```

tm_map.DCorpus

Transformations on Distributed Corpora

Description

Interface to apply transformation functions to distributed corpora. See [tm_map](#) in **tm** for more information.

Usage

```
## S3 method for class 'DCorpus'
tm_map(x, FUN, ...)
```

Arguments

x	A distributed corpus of class <code>DCorpus</code> .
FUN	a transformation function taking a text document as input and returning a text document. The function content_transformer can be used to create a wrapper to get and set the content of text documents.
...	arguments to FUN.

Value

A `DCorpus` with FUN applied to each document in x. If revisions are enabled, the original documents contained in x can be retrieved via getting back to the corresponding revision using the function `setRevision()`.

See Also

[getTransformations](#) for available transformations in package **tm**.

Examples

```
data("crude")  
tm_map(as.DCorpus(crude), content_transformer(tolower))
```

Index

as.DCorpus (DistributedCorpus), [2](#)
as.DistributedCorpus
 (DistributedCorpus), [2](#)
as.VCorpus.DCorpus (DistributedCorpus),
 [2](#)

content_transformer, [5](#)
Corpus, [2](#), [3](#)

DCorpus (DistributedCorpus), [2](#)
DirSource, [2](#)
DistributedCorpus, [2](#)
DList, [2](#)

getRevisions (Revisions), [3](#)
getTransformations, [5](#)

keepRevisions, [2](#)
keepRevisions (Revisions), [3](#)
keepRevisions<- (Revisions), [3](#)

language, [2](#)

removeRevision (Revisions), [3](#)
Revisions, [3](#)

setRevision (Revisions), [3](#)
Source, [2](#)

TermDocumentMatrix, [5](#)
TermDocumentMatrix.DCorpus, [4](#)
termFreq, [4](#), [5](#)
tm_map, [5](#)
tm_map.DCorpus, [5](#)