Package 'textmining'

September 26, 2016

Version 0.0.1

Date 2016-09-24

Title Integration of Text Mining and Topic Modeling Packages

Maintainer Jan Idziak <JanIdziak@gmail.com>

Description A framework for text mining and topic modelling. It provides an easy interface for using different topic modeling methods within R, by integrating the already existing packages. Full functionality of the package requires a local installation of 'TreeTagger'.

Suggests testthat, stringi, LDAvis

License GPL-3

RoxygenNote 5.0.1

Imports mallet, rJava, networkD3, wordcloud, dplyr, koRpus, topicmodels, slam, methods, SnowballC

Depends stylo, tm, NLP, caret

NeedsCompilation no

Author Jan Idziak [cre],
Maciej Eder [aut],
Tomasz Melcer [aut],
Andrew Goldstone [cph]

Repository CRAN

Date/Publication 2016-09-26 00:56:23

R topics documented:

`as.tmCorpus` *Create textmining Corpus*

Description

Create textmining Corpus

Usage

```
as.tmCorpus(x, ...)
```

Arguments

`x`	source object, can be tmTaggedText VCorpus or stylo.corpus.
`...`	metadata to be added

Examples

```
tmCorpus(c("This is new corpus", "And I like It"))
as.tmCorpus(c("This is new corpus", "And I like It"))
```

filter_documents *Function to filter tagged text*

Description

Function to filter tagged text

Usage

```
filter_documents(x, column, value)
```

Arguments

x	tmTaggedCorpus
column	column name
value	filtered value

Examples

```
## Not run:
library(dplyr)
library(textmining)
corp <- tmCorpus(c("This is corp corp", "Document 2 corp corp"))
rd <- tmTaggedCorpus(corp)
filtered_tmTaggedCorpus <- filter_documents(rd, "tag", "NP")
corpus <- tmCorpus(filtered_tmTaggedCorpus)

## End(Not run)
```

getDoc *Function to access documents for textmining objects*

Description

Function to access documents for textmining objects

Usage

```
getDoc(x, i = 1)
```

Arguments

x	object to retrive document fe tmCorpus or tmTextDocument
i	index

Value

returns i-th document of object x

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
getDoc(corp, 1)
getDoc(corp, 2)
content(corp)

text <- tmTextDocument("Text document new")
getDoc(text)
content(text)

parsed <- tmParsed(list(c("Parsed", "doc", "one"), c("Parsed", "two")))
getDoc(parsed, 2)
content(parsed)
```

---

getMeta *Function to access meta data for textmining objects*

Description

Function to access meta data for textmining objects

Usage

```
getMeta(x, parameter, i = 1)
```

Arguments

x	textmining object tmCorpus or tmTextDocument
parameter	name of metadata to be extracted
i	index

Value

returns i-th document of corpus x

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
getMeta(corp, parameter = "language", 1)
getMeta(corp, "title", 2)
meta(corp, "title")

text <- tmTextDocument("Text document")
```

```
getMeta(text, "language")
meta(text, "title")

parsed <- tmParsed(list(c("Parsed", "doc", "one"), c("Parsed", "two")))
getMeta(parsed, parameter = "title",  2)
meta(parsed, "title")
```

`make_tabled` *Function to create tmWordCountsTable object from tmParsed*

Description

Function to create tmWordCountsTable object from tmParsed

Usage

```
make_tabled(x)
```

Arguments

`x`	tmParsed source

Value

returns tmWordCountsTable object

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
parsed <- parse(corp)
tabled <- make_tabled(parsed)
content(tabled)
```

`mallet_prepare` *Helper function to use mallet topic modelling with tmCorpus*

Description

Helper function to use mallet topic modelling with tmCorpus

Usage

```
mallet_prepare(doc)
```

Arguments

doc	single document

Value

returns named character vector

---

ngram *Function to create ngram docs*

Description

Function to create ngram docs

Usage

```
ngram(x, k = 1)
```

Arguments

x	tmCorpus object
k	number of words in ngram

Value

returns tmParsed object of ngrams

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
parsed_ngrams <- ngram(corp, k = 2)
tabled <- make_tabled(parsed_ngrams)
content(tabled)
```

parse *Function to parse tmCorpus. As an outpus we have tmParsed object.*

Description

Function to parse tmCorpus. As an outpus we have tmParsed object.

Usage

```
parse(x)
```

Arguments

x	tmCorpus object

Value

returns tmParsed object

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
parsed <- parse(corp)
content(parsed)
```

predict *predict for* tmTopicModel *object*

Description

#' Function to predict topic model probabilities for existing topic model #' #' @param topic.model tmTopicModel obiect #' @param x tmCorpus object #' @param stoplist_file directory of file with stopwords #' @param token_regexp regular expression patterns #' @param burn_in parameter of mallet model #' @param sampling_interval parameter of mallet model #' @param n_iterations parameter of mallet model #' @param random_seed parameter of mallet model #' @param change predict structure so it fits normal #' #' @return returns the table of topic probabilities #' #' @export predict <- function(topic.model, x, stoplist_file = "en.txt", token_regexp = regexp_token, n_iterations = 100, sampling_interval = 10, burn_in = 10, random_seed = NULL) UseMethod("predict")

Function to predict topic model probabilities for an existing topic model. The code snippets for Mallet interface were derived from Andrew Goldstone's solution, posted at https://gist.github.com/agoldst/edcfd45b5ac371296b7

Usage

```
## S3 method for class 'tmTopicModel'
predict(object, x, stoplist_file = "en.txt",
  token_regexp = regexp_token, n_iterations = 100, sampling_interval = 10,
  burn_in = 10, random_seed = NULL, ...)

## S3 method for class 'LDA'
predict(object, x, ...)

## S3 method for class 'jobjRef'
predict(object, x, stoplist_file = "en.txt",
  token_regexp = regexp_token, n_iterations = 100, sampling_interval = 10,
  burn_in = 10, random_seed = NULL, ...)
```

Arguments

object	A tmTopicModel or LDA or jobjRef object
x	new data to predict probabilities of topics
stoplist_file	file direcroty or vector of stopwords
token_regexp	regular expression token
n_iterations	mallet LDA topic model parameter
sampling_interval	mallet LDA topic model parameter
burn_in	mallet LDA topic model parameter
random_seed	random seed
...	other motdel arguments

References

https://gist.github.com/agoldst/edcfd45b5ac371296b76

---

setDoc *Function to change documents for textmining objects*

Description

Function to change documents for textmining objects

Usage

```
setDoc(x, doc, i = 1)
```

Arguments

x	text mining object tmCorpus or tmTextDocument
doc	element to be attached as content
i	index

Value

x with changed i-th document

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
corp <- setDoc(corp, "Changed doc", 1)
getDoc(corp, 1)
content(corp) <- c("Content 1", "Content 2")
content(corp)

text <- tmTextDocument("Text document new")
text <- setDoc(text, "Content no 1")
getDoc(text)
content(text)

parsed <- tmParsed(list(c("Parsed", "doc", "one"), c("Parsed", "two")))
parsed <- setDoc(parsed, c("Changed", "document"), 2)
getDoc(parsed, 2)
content(parsed) <- list(c("Changed", "document", "one"), c("Changed", "two"))
content(parsed)
```

---

setMeta *Function to access meta data for textmining objects*

Description

Function to access meta data for textmining objects

Usage

```
setMeta(x, parameter, value, i = 1)
```

Arguments

x	object to retrive document fe tmCorpus or tmTextDocument
parameter	name of metadata to be extracted
value	value of meta data to set
i	index

Value

returns i-th document of corpus x

Examples

```
corp <- tmCorpus(c("This is first document", "This is second"))
corp <- setMeta(corp, parameter = "language", "pl", 1)
getMeta(corp, parameter = "language", 1)
meta(corp, "title") <- c("T1","T2")
meta(corp, "title")

text <- tmTextDocument("Text document")
text <- setMeta(text, "language", "de")
getMeta(text, "language")
meta(text, "title") <- "New"
meta(text, "title")

parsed <- tmParsed(list(c("Parsed", "doc", "one"), c("Parsed", "two")))
parsed <- setMeta(parsed, parameter = "title",  "New1", 2)
getMeta(parsed, parameter = "title",  2)
meta(parsed, "title") <- c("T1", "T2")
meta(parsed, "title")
```

---

tabler *Helper function for tabelarising documents*

Description

Helper function for tabelarising documents

Usage

```
tabler(x)
```

Arguments

x	parsed tmTextDocument object.

Value

returns tmTextDocument with table of words counts

terms *Function to return the most frequent terms of tmTopicModels*

Description

Function to return the most frequent terms of tmTopicModels

Usage

```
terms(x, ...)

## S3 method for class 'tmTopicModel'
terms(x, ...)

## S3 method for class 'jobjRef'
terms(x, ...)
```

Arguments

x	tmTopicModel
...	addictional parameters fe k - number of terms per topic

Value

data.frame with terms for topics

tmCorpus *Function to create tmCorpus*

Description

Function to create tmCorpus

Usage

```
tmCorpus(x = NULL, ..., package = "base")
```

Arguments

x	source, for package stylo it supposed to be directory with Corpus files
...	other arguments dependent on package that we use
package	package of reading the data could be tm or stylo

Value

returns tmCorpus object

Examples

```
corp <- tmCorpus(c("This is corp", "Document 2"))
corp <- tmCorpus(list("This is corp", "Document 2"))
corp <- tmCorpus(VectorSource(c("This is corp", "Document 2")), package = "tm")
## Not run:
corp <- tmCorpus(DirSource("directory"), package = "tm")
corp <- tmCorpus("directory", package = "stylo")

## End(Not run)
```

---

tmMetaData *Function to create tmMetaData*

Description

Function to create tmMetaData

Usage

```
tmMetaData(id = 1, language = "en", author = character(0),
  date = Sys.Date(), title = paste("Document_", id, sep = ""), ...)
```

Arguments

id	id of document
language	string language of document
author	string authors name
date	Date - date of reading in/date of publication
title	string title of document
...	other metadata

Value

returns tmMetaData object

Examples

```
tmMetaData(id = 1, author = "Author", newmetadata = "random")
tmMetaData(title = "New title")
```

tmParsed *Function to create tmParsed*

Description

Function to create tmParsed

Usage

```
tmParsed(x = NULL, package = "base", ...)
```

Arguments

x	source
package	package of reading the data could be tm or stylo
...	metadata for the tmParsed document

Value

returns tmParsed object

Examples

```
corp <- tmCorpus(c("This is corp", "Document 2"))
parsed <- parse(corp)
parsed_ngram <- ngram(corp, k = 2)
parsed <- tmParsed(list(c("This", "is", "corp"), c("Document", "2")))
## Not run:
corp <- tmCorpus("directory", package = "stylo")
parsed <- parse(corp)
parsed_ngram <- ngram(corp, k = 2)
parsed <- tmParsed(source = "directory", package = "stylo")

## End(Not run)
```

tmTaggedCorpus *Function to create tmTaggedCorpus*

Description

Function to create tmTaggedCorpus

Usage

```
tmTaggedCorpus(x = NULL, ..., treetagger = "manual", lang = "en",
  path = "C:\\TreeTagger", preset = "en")
```

Arguments

x	source, for package stylo it supposed to be directory with Corpus files
...	metadata
treetagger	treetagger settings
lang	language
path	treetagger path
preset	language seting

Value

returns tmTaggedCorpus object

Examples

```
## Not run:
corp <- tmCorpus(c("This is corp", "Document 2"))
tg_corp <- tmTaggedCorpus(corp)

## End(Not run)
```

---

tmTextDocument *Function to create single tmTextDocument with meta data. The object can store any from of documents: raw (string), parsed or table of words counts.*

Description

Function to create single tmTextDocument with meta data. The object can store any from of documents: raw (string), parsed or table of words counts.

Usage

```
tmTextDocument(x = NULL, ...)
```

Arguments

x	source
...	metadata to set. Can be set as language = "pl" or newmeta = "random"

Value

returns tmTextDocument

Examples

```
text <-  tmTextDocument("This is text")
text2 <- tmTextDocument("This is text", language = "en", title = "My test")
```

tmWordCountsTable *Function to create tmWordCountsTable*

Description

Function to create tmWordCountsTable

Usage

```
tmWordCountsTable(x = NULL)
```

Arguments

x	source tmParsed

Value

returns tmWordCountsTable object

Examples

```
corp <- tmCorpus(c("This is corp", "Document two is this"))
parsed <- parse(corp)
parsed_ngram <- ngram(corp, k = 2)
tabled <- make_tabled(parsed)
tabled_ngram <- make_tabled(parsed_ngram)
## Not run:
corp <- tmCorpus("directory", package = "stylo")
parsed <- parse(corp)
parsed_ngram <- ngram(corp, k = 2)
parsed <- tmParsed(source = "directory", package = "stylo")

## End(Not run)
```

topic_network *Function to plot topic network*

Description

Function to plot topic network

Usage

```
topic_network(k, model)
```

Arguments

k	Number of words from each topic to be included in graph
model	trained tmTopicModel object

Value

network The graph visualising the network

Examples

```
## Not run:
x <- tmCorpus(rep("as, a , a ,s  l k l l k k j h g f f hg j aaa", 100))
require(rJava)
model <- suppressMessages(train(x))
table_topic <- topic_table(model)
network <- topic_network(10 ,table_topic$words)

## End(Not run)
```

---

topic_table *Function to calculate topics and words arrays from the mallet model.*

Description

Function to calculate topics and words arrays from the mallet model.

Usage

```
topic_table(model)
```

Arguments

model	tmTopicModel mallet type model.

Value

topics Array of the topics.

words Array of the most important words in topic.

Examples

```
## Not run:
library(rJava)
x <- tmCorpus(lapply(1:100, function(x) paste(sample(LETTERS, 11),
                                              collapse = "")))

model <- train(x)
new_x <- tmCorpus(lapply(1:100, function(x) paste(sample(LETTERS, 11),
```

```
                                                  collapse = "")))

topic_table(model)

y <- DocumentTermMatrix(x)
rownames(y) <- meta(x, "title")
jss_TM <-
  list(VEM = train(y, k = k, control = list(seed = SEED)),
       VEM_fixed = train(y, k = k,
                         control = list(estimate.alpha = FALSE, seed = SEED)),
       Gibbs = train(y, k = k, method = "Gibbs",
                     control = list(seed = SEED, burnin = 1000,
                                    thin = 100, iter = 1000)))
pred_VEM <- predict(jss_TM$VEM, new_x)

## End(Not run)
```

---

topic_wordcloud *Simple wordcloud visualization of the topics.*

Description

Simple wordcloud visualization of the topics.

Usage

```
topic_wordcloud(model, topic_id = 1, k = 10, rot_per = 0,
  random_order = FALSE)
```

Arguments

model	tmTopicModel object
topic_id	Id of the analised topic.
k	number of words to be ploted.
rot_per	wordcloud param
random_order	order of words

Examples

```
## Not run:
library(rJava)
x <- tmCorpus(lapply(1:100, function(x) paste(sample(LETTERS, 11),
                                              collapse = "")))

model <- train(x)
topic_wordcloud(model, topic_id = 2, k = 11)

## End(Not run)
```

`train` *train for* `tmCorpus` *object*

Description

#' Function to create train Topic Model #' #' @param x tmCorpus object #' @param ... settings for mallet.doc.topics and mallet.topic.words #' #' @return returns object of a class tmTopicModel #' #' @export train <- function(x, ...) UseMethod("train")

Usage

```
## S3 method for class 'tmCorpus'
train(x, k = 20, stoplist_file = "en.txt",
  token_regexp = regexp_token, alpha_opt = 20, burn_in = 50,
  train = 200, maximize = 10, package = "mallet", ...)

## S3 method for class 'DocumentTermMatrix'
train(x, k = 20, ...)
```

Arguments

`x`	A `tmCorpus` object or `DocumentTermMatrix` object
`k`	number of topics
`stoplist_file`	file direcroty or vector of stopwords
`token_regexp`	regular expression token
`alpha_opt`	mallet LDA topic model parameter
`burn_in`	mallet LDA topic model parameter
`train`	mallet LDA topic model parameter
`maximize`	mallet LDA topic model parameter
`package`	package to train topic mdoel can be set to "mallet" or "topicmodels"
`...`	other model arguments

Index