Package ‘textmineR’

June 28, 2021

Type Package

Title Functions for Text Mining and Topic Modeling

Version 3.0.5

Description An aid for text mining in R, with a syntax that
should be familiar to experienced R users. Provides a wrapper for several
topic models that take similarly-formatted input and give similarly-formatted
output. Has additional functionality for analyzing and diagnostics for
topic models.

SystemRequirements GNU make, C++11

Depends R (>= 3.0.2), Matrix

Imports gtools, magrittr, methods, parallel, text2vec (>= 0.5),
stopwords, stringr, Rcpp, RcppProgress, RSpectra, utils

Suggests spelling, digest, dplyr, igraph, knitr, lda, MASS, rmarkdown,
SnowballC, stringi, testthat, tibble, tidyr, tidytext,
topicmodels, wordcloud

License MIT + file LICENSE

URL `https://www.rtextminer.com/`

BugReports `https://github.com/TommyJones/textmineR/issues`

LazyData true

LinkingTo Rcpp, RcppArmadillo, RcppProgress

RoxygenNote 7.1.1

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Author Tommy Jones [aut, cre],
William Doane [ctb],
Mattias Attbom [ctb]

Maintainer Tommy Jones `<jones.thos.w@gmail.com>`

Repository CRAN

Date/Publication 2021-06-28 05:00:02 UTC

R topics documented:

CalcGamma *Calculate a matrix whose rows represent P(topic_i|tokens)*

Description

This function takes a phi matrix (P(token|topic)) and a theta matrix (P(topic|document)) and returns the phi prime matrix (P(topic|token)). Phi prime can be used for classifying new documents and for alternative topic labels.

Usage

```
CalcGamma(phi, theta, p_docs = NULL, correct = TRUE)
```

Arguments

`phi` The phi matrix whose rows index topics and columns index words. The i, j entries are P(word_i | topic_j)

`theta` The theta matrix whose rows index documents and columns index topics. The i, j entries are P(topic_i | document_j)

`p_docs` A numeric vector of length `nrow(theta)` that is proportional to the number of terms in each document. This is an optional argument. It defaults to NULL

`correct` Logical. Do you want to set NAs or NaNs in the final result to zero? Useful when hitting computational underflow. Defaults to `TRUE`. Set to `FALSE` for troubleshooting or diagnostics.

Value

Returns a `matrix` whose rows correspond to topics and whose columns correspond to tokens. The i,j entry corresponds to P(topic_i|token_j)

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_topic_model)

# Make a gamma matrix, P(topic|words)
gamma <- CalcGamma(phi = nih_sample_topic_model$phi,
                   theta = nih_sample_topic_model$theta)
```

`CalcHellingerDist` *Calculate Hellinger Distance*

Description

Calculates the Hellinger distances or the rows or columns of a numeric matrix or for two numeric vectors.

Usage

```
CalcHellingerDist(x, y = NULL, by_rows = TRUE)
```

Arguments

`x` A numeric matrix or numeric vector

`y` A numeric vector. `y` must be specified if `x` is a numeric vector.

`by_rows` Logical. If `x` is a matrix, should distances be calculated by rows?

Value

If x is a matrix, this returns an square and symmetric matrix. The i,j entries correspond to the Hellinger Distance between the rows of x (or the columns of x if by_rows = FALSE). If x and y are vectors, this returns a numeric scalar whose value is the Hellinger Distance between x and y.

Examples

```
x <- rchisq(n = 100, df = 8)
y <- x^2
CalcHellingerDist(x = x, y = y)

mymat <- rbind(x, y)
CalcHellingerDist(x = mymat)
```

CalcJSDivergence *Calculate Jensen-Shannon Divergence*

Description

This function calculates the Jensen Shannon Divergence for the rows or columns of a numeric matrix or for two numeric vectors.

Usage

```
CalcJSDivergence(x, y = NULL, by_rows = TRUE)
```

Arguments

x	A numeric matrix or numeric vector
y	A numeric vector. y must be specified if x is a numeric vector.
by_rows	Logical. If x is a matrix, should distances be calculated by rows?

Value

If x is a matrix, this returns an square and symmetric matrix. The i,j entries correspond to the Hellinger Distance between the rows of x (or the columns of x if by_rows = FALSE). If x and y are vectors, this returns a numeric scalar whose value is the Hellinger Distance between x and y.

Examples

```
x <- rchisq(n = 100, df = 8)
y <- x^2
CalcJSDivergence(x = x, y = y)

mymat <- rbind(x, y)
CalcJSDivergence(x = mymat)
```

CalcLikelihood *Calculate the log likelihood of a document term matrix given a topic model*

Description

This function takes a DTM, phi matrix (P(word|topic)), and a theta matrix (P(topic|document)) and returns a single value for the likelihood of the data given the model.

Usage

```
CalcLikelihood(dtm, phi, theta, ...)
```

Arguments

`dtm` The document term matrix of class `dgCMatrix`.

`phi` The phi matrix whose rows index topics and columns index words. The i, j entries are P(word_i | topic_j)

`theta` The theta matrix whose rows index documents and columns index topics. The i, j entries are P(topic_i | document_j)

`...` Other arguments to pass to `TmParallelApply`. See note, below.

Value

Returns an object of class `numeric` corresponding to the log likelihood.

Note

This function performs parallel computation if `dtm` has more than 3,000 rows. The default is to use all available cores according to `detectCores`. However, this can be modified by passing the `cpus` argument when calling this function.

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_dtm)
data(nih_sample_topic_model)

# Get the likelihood of the data given the fitted model parameters
ll <- CalcLikelihood(dtm = nih_sample_dtm,
                     phi = nih_sample_topic_model$phi,
                     theta = nih_sample_topic_model$theta)

ll
```

CalcProbCoherence *Probabilistic coherence of topics*

Description

Calculates the probabilistic coherence of a topic or topics. This approximates semantic coherence or human understandability of a topic.

Usage

```
CalcProbCoherence(phi, dtm, M = 5)
```

Arguments

phi	A numeric matrix or a numeric vector. The vector, or rows of the matrix represent the numeric relationship between topic(s) and terms. For example, this relationship may be p(word|topic) or p(topic|word).
dtm	A document term matrix or co-occurrence matrix of class `matrix` or whose class inherits from the `Matrix` package. Columns must index terms.
M	An integer for the number of words to be used in the calculation. Defaults to 5

Value

Returns an object of class `numeric` corresponding to the probabilistic coherence of the input topic(s).

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_topic_model)
data(nih_sample_dtm)

CalcProbCoherence(phi = nih_sample_topic_model$phi, dtm = nih_sample_dtm, M = 5)
```

CalcTopicModelR2 *Calculate the R-squared of a topic model.*

Description

Function to calculate R-squared for a topic model. This uses a geometric interpretation of R-squared as the proportion of total distance each document is from the center of all the documents that is explained by the model.

Usage

```
CalcTopicModelR2(dtm, phi, theta, ...)
```

Arguments

`dtm` A documents by terms dimensional document term matrix of class `dgCMatrix` or of class `matrix`.

`phi` A topics by terms dimensional matrix where each entry is p(term_i |topic_j)

`theta` A documents by topics dimensional matrix where each entry is p(topic_j|document_d)

`...` Other arguments to be passed to `TmParallelApply`. See note, below.

Value

Returns an object of class `numeric` representing the proportion of variability in the data that is explained by the topic model.

Note

This function performs parallel computation if `dtm` has more than 3,000 rows. The default is to use all available cores according to `detectCores`. However, this can be modified by passing the cpus argument when calling this function.

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_dtm)
data(nih_sample_topic_model)

# Get the R-squared of the model
r2 <- CalcTopicModelR2(dtm = nih_sample_dtm,
                     phi = nih_sample_topic_model$phi,
                     theta = nih_sample_topic_model$theta)


r2
```

`Cluster2TopicModel` *Represent a document clustering as a topic model*

Description

Represents a document clustering as a topic model of two matrices. phi: P(term | cluster) theta: P(cluster | document)

Usage

```
Cluster2TopicModel(dtm, clustering, ...)
```

Arguments

dtm	A document term matrix of class dgCMatrix or whose class inherits from the Matrix package. Columns must index terms, rows must index documents.
clustering	A vector of length nrow(dtm) whose entries form a partitional clustering of the documents.
...	Other arguments to be passed to TmParallelApply.

Value

Returns a list with two elements, phi and theta. 'phi' is a matrix whose j-th row represents P(terms | cluster_j). 'theta' is a matrix whose j-th row represents P(clusters | document_j). Each row of theta should only have one non-zero element.

Examples

```
## Not run:
# Load pre-formatted data for use
data(nih_sample_dtm)
data(nih_sample)

result <- Cluster2TopicModel(dtm = nih_sample_dtm,
                             clustering = nih_sample$IC_NAME)

## End(Not run)
```

CreateDtm *Convert a character vector to a document term matrix.*

Description

This is the main document term matrix creating function for textmineR. In most cases, all you need to do is import documents as a character vector in R and then run this function to get a document term matrix that is compatible with the rest of textmineR's functionality and many other libraries. CreateDtm is built on top of the excellent text2vec library.

Usage

```
CreateDtm(
  doc_vec,
  doc_names = names(doc_vec),
  ngram_window = c(1, 1),
 stopword_vec = c(stopwords::stopwords("en"), stopwords::stopwords(source = "smart")),
  lower = TRUE,
  remove_punctuation = TRUE,
  remove_numbers = TRUE,
  stem_lemma_function = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`doc_vec` A character vector of documents.

`doc_names` A vector of names for your documents. Defaults to `names(doc_vec)`. If NULL, then doc_names is set to be `1:length(doc_vec)`.

`ngram_window` A numeric vector of length 2. The first entry is the minimum n-gram size; the second entry is the maximum n-gram size. Defaults to `c(1,1)`.

`stopword_vec` A character vector of stopwords you would like to remove. Defaults to `c(stopwords::stopwords("en") = "smart"))`. If you do not want stopwords removed, specify `stopword_vec = c()`.

`lower` Do you want all words coerced to lower case? Defaults to `TRUE`

`remove_punctuation` Do you want to convert all non-alpha numeric characters to spaces? Defaults to `TRUE`

`remove_numbers` Do you want to convert all numbers to spaces? Defaults to `TRUE`

`stem_lemma_function` A function that you would like to apply to the documents for stemming, lemmatization, or similar. See examples for usage.

`verbose` Defaults to `TRUE`. Do you want to see status during vectorization?

`...` Other arguments to be passed to `TmParallelApply`.

Value

A document term matrix of class `dgCMatrix`. The rows index documents. The columns index terms. The i, j entries represent the count of term j appearing in document i.

Note

The following transformations are applied to `stopword_vec` as well as `doc_vec`: `lower`, `remove_punctuation`, `remove_numbers`

See `stopwords` for details on the default to the `stopword_vec` argument.

Examples

```
## Not run:
data(nih_sample)

# DTM of unigrams and bigrams
dtm <- CreateDtm(doc_vec = nih_sample$ABSTRACT_TEXT,
                 doc_names = nih_sample$APPLICATION_ID,
                 ngram_window = c(1, 2))

# DTM of unigrams with Porter's stemmer applied
dtm <- CreateDtm(doc_vec = nih_sample$ABSTRACT_TEXT,
                 doc_names = nih_sample$APPLICATION_ID,
                 stem_lemma_function = function(x) SnowballC::wordStem(x, "porter"))

## End(Not run)
```

CreateTcm *Convert a character vector to a term co-occurrence matrix.*

Description

This is the main term co-occurrence matrix creating function for `textmineR`. In most cases, all you need to do is import documents as a character vector in R and then run this function to get a term co-occurrence matrix that is compatible with the rest of `textmineR`'s functionality and many other libraries. `CreateTcm` is built on top of the excellent `text2vec` library.

Usage

```
CreateTcm(
  doc_vec,
  skipgram_window = Inf,
  ngram_window = c(1, 1),
 stopword_vec = c(stopwords::stopwords("en"), stopwords::stopwords(source = "smart")),
  lower = TRUE,
  remove_punctuation = TRUE,
  remove_numbers = TRUE,
  stem_lemma_function = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`doc_vec` A character vector of documents.

`skipgram_window` An integer window, from `0` to `Inf` for skip-grams. Defaults to `Inf`. See 'Details', below.

`ngram_window` A numeric vector of length 2. The first entry is the minimum n-gram size; the second entry is the maximum n-gram size. Defaults to `c(1,1)`. Must be `c(1,1)` if `skipgram_window` is not `0` or `Inf`.

`stopword_vec` A character vector of stopwords you would like to remove. Defaults to `c(stopwords::stopwords("en")` `= "smart"))`. If you do not want stopwords removed, specify `stopword_vec = c()`.

`lower` Do you want all words coerced to lower case? Defaults to `TRUE`

`remove_punctuation` Do you want to convert all non-alpha numeric characters to spaces? Defaults to `TRUE`

`remove_numbers` Do you want to convert all numbers to spaces? Defaults to `TRUE`

`stem_lemma_function` A function that you would like to apply to the documents for stemming, lemmatization, or similar. See examples for usage.

`verbose` Defaults to `TRUE`. Do you want to see status during vectorization?

`...` Other arguments to be passed to `TmParallelApply`.

Details

Setting skipgram_window counts the number of times that term j appears within skipgram_window places of term i. Inf and 0 create somewhat special TCMs. Setting skipgram_window to Inf counts the number of documents in which term j and term i occur together. Setting skipgram_window to 0 counts the number of terms shared by document j and document i. A TCM where skipgram_window is 0 is the only TCM that will be symmetric.

Value

A document term matrix of class dgCMatrix. The rows index documents. The columns index terms. The i, j entries represent the count of term j appearing in document i.

Note

The following transformations are applied to stopword_vec as well as doc_vec: lower, remove_punctuation, remove_numbers

See stopwords for details on the default to the stopword_vec argument.

Examples

```
## Not run:
data(nih_sample)

# TCM of unigrams and bigrams
tcm <- CreateTcm(doc_vec = nih_sample$ABSTRACT_TEXT,
                 skipgram_window = Inf,
                 ngram_window = c(1, 2))

# TCM of unigrams and a skip=gram window of 3, applying Porter's word stemmer
tcm <- CreateTcm(doc_vec = nih_sample$ABSTRACT_TEXT,
                 skipgram_window = 3,
                 stem_lemma_function = function(x) SnowballC::wordStem(x, "porter"))

## End(Not run)
```

Dtm2Docs *Convert a DTM to a Character Vector of documents*

Description

This function takes a sparse matrix (DTM) as input and returns a character vector whose length is equal to the number of rows of the input DTM.

Usage

```
Dtm2Docs(dtm, ...)
```

Arguments

`dtm` A sparse Matrix from the matrix package whose rownames correspond to documents and colnames correspond to words

`...` Other arguments to be passed to `TmParallelApply`. See note, below.

Value

Returns a character vector. Each entry of this vector corresponds to the rows of `dtm`.

Note

This function performs parallel computation if `dtm` has more than 3,000 rows. The default is to use all available cores according to `detectCores`. However, this can be modified by passing the `cpus` argument when calling this function.

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample)
data(nih_sample_dtm)

# see the original documents
nih_sample$ABSTRACT_TEXT[ 1:3 ]

# see the new documents re-structured from the DTM
new_docs <- Dtm2Docs(dtm = nih_sample_dtm)

new_docs[ 1:3 ]
```

`Dtm2Lexicon` *Turn a document term matrix into a list for LDA Gibbs sampling*

Description

Represents a document term matrix as a list.

Usage

```
Dtm2Lexicon(dtm, ...)
```

Arguments

`dtm` A document term matrix (or term co-occurrence matrix) of class `dgCMatrix`.

`...` Other arguments to be passed to `TmParallelApply`.

Value

Returns a list. Each element of the list represents a row of the input matrix. Each list element contains a numeric vector with as many entries as tokens in the original document. The entries are the column index for that token, minus 1.

Examples

```
## Not run:
# Load pre-formatted data for use
data(nih_sample_dtm)

result <- Dtm2Lexicon(dtm = nih_sample_dtm,
                      cpus = 2)

## End(Not run)
```

`Dtm2Tcm` *Turn a document term matrix into a term co-occurrence matrix*

Description

Turn a document term matrix, whose rows index documents and whose columns index terms, into a term co-occurrence matrix. A term co-occurrence matrix's rows and columns both index terms. See `details`, below.

Usage

```
Dtm2Tcm(dtm)
```

Arguments

`dtm`	A document term matrix, generally of class `dgCMatrix`, though other classes, such as `dgTMatrix`, may also work without issue.

Value

Returns a square `dgCMatrix` whose rows and columns both index terms. The i, j entries of this matrix represent the count of term j across documents containing term i. Note that, while square, this matrix is not symmetric.

Examples

```
data(nih_sample_dtm)

tcm <- Dtm2Tcm(nih_sample_dtm)
```

`FitCtmModel` *Fit a Correlated Topic Model*

Description

A wrapper for the CTM function based on Blei's original code that returns a nicely-formatted topic model.

Usage

```
FitCtmModel(
  dtm,
  k,
  calc_coherence = TRUE,
  calc_r2 = FALSE,
  return_all = TRUE,
  ...
)
```

Arguments

`dtm` A document term matrix of class `dgCMatrix`

`k` Number of topics

`calc_coherence` Do you want to calculate probabilistic coherence of topics after the model is trained? Defaults to `TRUE`.

`calc_r2` Do you want to calculate R-squared after the model is trained? Defaults to `FALSE`.

`return_all` Logical. Do you want the raw results of the underlying function returned along with the formatted results? Defaults to `TRUE`.

`...` Other arguments to pass to CTM or TmParallelApply. See note below.

Value

Returns a list with a minimum of two objects, `phi` and `theta`. The rows of `phi` index topics and the columns index tokens. The rows of `theta` index documents and the columns index topics.

Note

When passing additional arguments to CTM, you must unlist the elements in the `control` argument and pass them one by one. See examples for how to dot this correctly.

Examples

```
# Load a pre-formatted dtm
data(nih_sample_dtm)

# Fit a CTM model on a sample of documents
model <- FitCtmModel(dtm = nih_sample_dtm[ sample(1:nrow(nih_sample_dtm) , 10) , ],
                     k = 3, return_all = FALSE)

# the correct way to pass control arguments to CTM
## Not run:
topics_CTM <- FitCtmModel(
    dtm = nih_sample_dtm[ sample(1:nrow(nih_sample_dtm) , 10) , ],
    k = 10,
    calc_coherence = TRUE,
    calc_r2 = TRUE,
    return_all = TRUE,
    estimate.beta = TRUE,
    verbose = 0,
    prefix = tempfile(),
    save = 0,
    keep = 0,
    seed = as.integer(Sys.time()),
    nstart = 1L,
    best = TRUE,
    var = list(iter.max = 500, tol = 10^-6),
    em = list(iter.max = 1000, tol = 10^-4),
    initialize = "random",
    cg = list(iter.max = 500, tol = 10^-5)
)

## End(Not run)
```

FitLdaModel *Fit a Latent Dirichlet Allocation topic model*

Description

Fit a Latent Dirichlet Allocation topic model using collapsed Gibbs sampling.

Usage

```
FitLdaModel(
  dtm,
  k,
  iterations = NULL,
  burnin = -1,
  alpha = 0.1,
  beta = 0.05,
  optimize_alpha = FALSE,
  calc_likelihood = FALSE,
  calc_coherence = TRUE,
  calc_r2 = FALSE,
  ...
)
```

Arguments

`dtm` A document term matrix or term co-occurrence matrix of class dgCMatrix

`k` Integer number of topics

`iterations` Integer number of iterations for the Gibbs sampler to run. A future version may include automatic stopping criteria.

`burnin` Integer number of burnin iterations. If `burnin` is greater than -1, the resulting "phi" and "theta" matrices are an average over all iterations greater than `burnin`.

`alpha` Vector of length `k` for asymmetric or a number for symmetric. This is the prior for topics over documents

`beta` Vector of length `ncol(dtm)` for asymmetric or a number for symmetric. This is the prior for words over topics.

`optimize_alpha` Logical. Do you want to optimize alpha every 10 Gibbs iterations? Defaults to `FALSE`.

`calc_likelihood` Do you want to calculate the likelihood every 10 Gibbs iterations? Useful for assessing convergence. Defaults to `FALSE`.

`calc_coherence` Do you want to calculate probabilistic coherence of topics after the model is trained? Defaults to `TRUE`.

`calc_r2` Do you want to calculate R-squared after the model is trained? Defaults to `FALSE`.

`...` Other arguments to be passed to `TmParallelApply`

Details

EXPLAIN IMPLEMENTATION DETAILS

Value

Returns an S3 object of class c("LDA", "TopicModel"). DESCRIBE MORE

Examples

```
# load some data
data(nih_sample_dtm)

# fit a model
set.seed(12345)
m <- FitLdaModel(dtm = nih_sample_dtm[1:20,], k = 5,
                 iterations = 200, burnin = 175)
```

```
str(m)

# predict on held-out documents using gibbs sampling "fold in"
p1 <- predict(m, nih_sample_dtm[21:100,], method = "gibbs",
              iterations = 200, burnin = 175)

# predict on held-out documents using the dot product method
p2 <- predict(m, nih_sample_dtm[21:100,], method = "dot")

# compare the methods
barplot(rbind(p1[1,],p2[1,]), beside = TRUE, col = c("red", "blue"))
```

FitLsaModel — *Fit a topic model using Latent Semantic Analysis*

Description

A wrapper for RSpectra::svds that returns a nicely-formatted latent semantic analysis topic model.

Usage

```
FitLsaModel(dtm, k, calc_coherence = TRUE, return_all = FALSE, ...)
```

Arguments

dtm	A document term matrix of class Matrix::dgCMatrix
k	Number of topics
calc_coherence	Do you want to calculate probabilistic coherence of topics after the model is trained? Defaults to TRUE.
return_all	Should all objects returned from RSpectra::svds be returned here? Defaults to FALSE
...	Other arguments to pass to svds through its opts parameter.

Details

Latent semantic analysis, LSA, uses single value decomposition to factor the document term matrix. In many LSA applications, TF-IDF weights are applied to the DTM before model fitting. However, this is not strictly necessary.

Value

Returns a list with a minimum of three objects: phi, theta, and sv. The rows of phi index topics and the columns index tokens. The rows of theta index documents and the columns index topics. sv is a vector of singular values.

Examples

```
# Load a pre-formatted dtm
data(nih_sample_dtm)

# Convert raw word counts to TF-IDF frequency weights
idf <- log(nrow(nih_sample_dtm) / Matrix::colSums(nih_sample_dtm > 0))

dtm_tfidf <- Matrix::t(nih_sample_dtm) * idf

dtm_tfidf <- Matrix::t(dtm_tfidf)

# Fit an LSA model
model <- FitLsaModel(dtm = dtm_tfidf, k = 5)

str(model)
```

GetProbableTerms *Get cluster labels using a "more probable" method of terms*

Description

Function extracts probable terms from a set of documents. Probable here implies more probable than in a corpus overall.

Usage

```
GetProbableTerms(docnames, dtm, p_terms = NULL)
```

Arguments

docnames	A character vector of rownames of dtm for set of documents
dtm	A document term matrix of class `matrix` or `dgCMatrix`.
p_terms	If not NULL (the default), a numeric vector representing the probability of each term in the corpus whose names correspond to colnames(dtm).

Value

Returns a numeric vector of the format p_terms. The entries of the vectors correspond to the difference in the probability of drawing a term from the set of documents given by docnames and the probability of drawing that term from the corpus overall (p_terms).

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_topic_model)
data(nih_sample_dtm)

# documents with a topic proportion of .25 or higher for topic 2
mydocs <- rownames(nih_sample_topic_model$theta)[ nih_sample_topic_model$theta[ , 2 ] >= 0.25 ]

term_probs <- Matrix::colSums(nih_sample_dtm) / sum(Matrix::colSums(nih_sample_dtm))

GetProbableTerms(docnames = mydocs, dtm = nih_sample_dtm, p_terms = term_probs)
```

GetTopTerms *Get Top Terms for each topic from a topic model*

Description

Takes topics by terms matrix and returns top M terms for each topic

Usage

```
GetTopTerms(phi, M, return_matrix = TRUE)
```

Arguments

phi	A matrix whose rows index topics and columns index words
M	An integer for the number of terms to return
return_matrix	Do you want a `matrix` or `data.frame/tibble` returned? Defaults to `TRUE`.

Value

If `return_matrix` = `TRUE` (the default) then a matrix. Otherwise, returns a `data.frame` or `tibble` whose columns correspond to a topic and whose m-th row correspond to the m-th top term from the input `phi`.

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_topic_model)

top_terms <- GetTopTerms(phi = nih_sample_topic_model$phi, M = 5)

str(top_terms)
```

Internals *Internal helper functions for* textmineR

Description

These functions are internal helper functions for textmineR. They are not designed to be called by users. Each of the functions here are C++ functions. There are corresponding R functions that call these that add additional functionality.

LabelTopics *Get some topic labels using a "more probable" method of terms*

Description

Function calls GetProbableTerms with some rules to get topic labels. This function is in "super-ultra-mega alpha"; use at your own risk/discretion.

Usage

```
LabelTopics(assignments, dtm, M = 2)
```

Arguments

assignments	A documents by topics matrix similar to theta. This will work best if this matrix is sparse, with only a few non-zero topics per document.
dtm	A document term matrix of class matrix or dgCMatrix. The columns of dtm should be n-grams whose colnames have a "_" where spaces would be between the words.
M	The number of n-gram labels you want to return. Defaults to 2

Value

Returns a matrix whose rows correspond to topics and whose j-th column corresponds to the j-th "best" label assignment.

Examples

```
# make a dtm with unigrams and bigrams
data(nih_sample_topic_model)

m <- nih_sample_topic_model

assignments <- t(apply(m$theta, 1, function(x){
  x[ x < 0.05 ] <- 0
  x / sum(x)
}))
```

```
assignments[is.na(assignments)] <- 0

labels <- LabelTopics(assignments = assignments, dtm = m$data, M = 2)
```

nih *Abstracts and metadata from NIH research grants awarded in 2014*

Description

This dataset holds information on research grants awarded by the National Institutes of Health (NIH) in 2014. The data set was downloaded in approximately January of 2015 from https://exporter.nih.gov/ExPORTER_Catalog.aspx. It includes both 'projects' and 'abstracts' files.

Usage

```
data("nih_sample")
data("nih_sample_dtm")
data("nih_sample_topic_model")
```

Format

A `data.frame` of 100 randomly-sampled grants' abstracts and metadata. A `dgCMatrix` representing the document term matrix of abstracts from 100 randomly-sampled grants. A `list` containing a topic model of these 100 sampled grants.

Source

National Institutes of Health ExPORTER https://exporter.nih.gov/ExPORTER_Catalog.aspx

posterior *Posterior methods for topic models*

Description

`posterior` will draw from the posterior distribution of a topic model

Usage

```
posterior(object, ...)
```

Arguments

`object`	An existing trained topic model
`...`	Additional arguments to the call

posterior.lda_topic_model

Draw from the posterior of an LDA topic model

Description

This function takes an object of class `lda_topic_model` and draws samples from the posterior of either `phi` or `theta`. This is useful for quantifying uncertainty around parametersof the final model.

Usage

```
## S3 method for class 'lda_topic_model'
posterior(object, which = "theta", num_samples = 100, ...)
```

Arguments

`object` An object of class `lda_topic_model`

`which` A character of either ’theta’ or ’phi’, indicating from which matrix to draw posterior samples

`num_samples` Integer number of samples to draw

`...` Other arguments to be passed to `TmParallelApply`.

Value

Returns a data frame where each row is a single sample from the posterior. Each column is the distribution over a single parameter. The variable `var` is a facet for subsetting by document (for theta) or topic (for phi).

References

Heinrich, G. (2005) Parameter estimation for text analysis. Technical report. http://www.arbylon.net/publications/text-est.pdf

Examples

```
## Not run:
a <- posterior(object = nih_sample_topic_model, which = "theta", num_samples = 20)

plot(density(a$t1[a$var == "8693991"]))

b <- posterior(object = nih_sample_topic_model, which = "phi", num_samples = 20)

plot(denisty(b$research[b$var == "t_5"]))

## End(Not run)
```

predict.ctm_topic_model *Predict method for Correlated topic models (CTM)*

Description

Obtains predictions of topics for new documents from a fitted CTM model

Usage

```
## S3 method for class 'ctm_topic_model'
predict(object, newdata, ...)
```

Arguments

`object`	a fitted object of class "ctm_topic_model"
`newdata`	a DTM or TCM of class dgCMatrix or a numeric vector
`...`	further arguments passed to or from other methods.

Value

a "theta" matrix with one row per document and one column per topic

Note

Predictions for this method are performed using the "dot" method as described in the textmineR vignette "c_topic_modeling".

Examples

```
# Load a pre-formatted dtm
## Not run:
data(nih_sample_dtm)

model <- FitCtmModel(dtm = nih_sample_dtm[1:20,], k = 3,
                     calc_coherence = FALSE, calc_r2 = FALSE)

# Get predictions on the next 50 documents
pred <- predict(model, nih_sample_dtm[21:100,])

## End(Not run)
```

predict.lda_topic_model

Get predictions from a Latent Dirichlet Allocation model

Description

Obtains predictions of topics for new documents from a fitted LDA model

Usage

```
## S3 method for class 'lda_topic_model'
predict(
  object,
  newdata,
  method = c("gibbs", "dot"),
  iterations = NULL,
  burnin = -1,
  ...
)
```

Arguments

`object` a fitted object of class `lda_topic_model`

`newdata` a DTM or TCM of class `dgCMatrix` or a numeric vector

`method` one of either "gibbs" or "dot". If "gibbs" Gibbs sampling is used and `iterations` must be specified.

`iterations` If `method = "gibbs"`, an integer number of iterations for the Gibbs sampler to run. A future version may include automatic stopping criteria.

`burnin` If `method = "gibbs"`, an integer number of burnin iterations. If `burnin` is greater than -1, the entries of the resulting "theta" matrix are an average over all iterations greater than `burnin`.

`...` Other arguments to be passed to `TmParallelApply`

Value

a "theta" matrix with one row per document and one column per topic

Examples

```
## Not run:
# load some data
data(nih_sample_dtm)

# fit a model
set.seed(12345)
```

```
m <- FitLdaModel(dtm = nih_sample_dtm[1:20,], k = 5,
                 iterations = 200, burnin = 175)

str(m)

# predict on held-out documents using gibbs sampling "fold in"
p1 <- predict(m, nih_sample_dtm[21:100,], method = "gibbs",
              iterations = 200, burnin = 175)

# predict on held-out documents using the dot product method
p2 <- predict(m, nih_sample_dtm[21:100,], method = "dot")

# compare the methods
barplot(rbind(p1[1,],p2[1,]), beside = TRUE, col = c("red", "blue"))

## End(Not run)
```

predict.lsa_topic_model

Predict method for LSA topic models

Description

Obtains predictions of topics for new documents from a fitted LSA model

Usage

```
## S3 method for class 'lsa_topic_model'
predict(object, newdata, ...)
```

Arguments

object	a fitted object of class "lsa_topic_model"
newdata	a DTM or TCM of class dgCMatrix or a numeric vector
...	further arguments passed to or from other methods.

Value

a "theta" matrix with one row per document and one column per topic

Examples

```
# Load a pre-formatted dtm
data(nih_sample_dtm)

# Convert raw word counts to TF-IDF frequency weights
idf <- log(nrow(nih_sample_dtm) / Matrix::colSums(nih_sample_dtm > 0))

dtm_tfidf <- Matrix::t(nih_sample_dtm) * idf
dtm_tfidf <- Matrix::t(dtm_tfidf)

# Fit an LSA model on the first 50 documents
model <- FitLsaModel(dtm = dtm_tfidf[1:50,], k = 5)

# Get predictions on the next 50 documents
pred <- predict(model, dtm_tfidf[51:100,])
```

SummarizeTopics *Summarize topics in a topic model*

Description

Create a data frame summarizing the contents of each topic in a model

Usage

```
SummarizeTopics(model)
```

Arguments

`model`	A list (or S3 object) with three named matrices: phi, theta, and gamma. These conform to outputs of many of textmineR's native topic modeling functions such as FitLdaModel.

Details

'prevalence' is normalized to sum to 100. If your 'theta' matrix has negative values (as may be the case with an LSA model), a constant is added so that the least prevalent topic has a prevalence of 0.

'coherence' is calculated using CalcProbCoherence.

'label' is assigned using the top label from LabelTopics. This requires an "assignment" matrix. This matrix is like a "theta" matrix except that it is binary. A topic is "in" a document or it is not. The assignment is made by comparing each value of theta to the minimum of the largest value for each row of theta (each document). This ensures that each document has at least one topic assigned to it.

Value

An object of class `data.frame` or `tibble` with 6 columns: 'topic' is the name of the topic, 'prevalence' is the rough prevalence of the topic in all documents across the corpus, 'coherence' is the probabilistic coherence of the topic, 'top_terms_phi' are the top 5 terms for each topic according to P(word|topic), 'top_terms_gamma' are the top 5 terms for each topic according to P(topic|word).

Examples

```
## Not run:
SummarizeTopics(nih_sample_topic_model)

## End(Not run)
```

TermDocFreq *Get term frequencies and document frequencies from a document term matrix.*

Description

This function takes a document term matrix as input and returns a data frame with columns for term frequency, document frequency, and inverse-document frequency

Usage

```
TermDocFreq(dtm)
```

Arguments

`dtm`	A document term matrix of class `dgCMatrix`.

Value

Returns a `data.frame` or `tibble` with 4 columns. The first column, `term` is a vector of token labels. The second column, `term_freq` is the count of times `term` appears in the entire corpus. The third column `doc_freq` is the count of the number of documents in which `term` appears. The fourth column, `idf` is the log-weighted inverse document frequency of `term`.

Examples

```
# Load a pre-formatted dtm and topic model
data(nih_sample_dtm)
data(nih_sample_topic_model)

# Get the term frequencies
term_freq_mat <- TermDocFreq(nih_sample_dtm)

str(term_freq_mat)
```

textmineR *textmineR*

Description

Functions for Text Mining and Topic Modeling

Details

An aid for text mining in R, with a syntax that should be familiar to experienced R users. Provides a wrapper for several topic models that take similarly-formatted input and give similarly-formatted output. Has additional functionality for analyzing and diagnostics for topic models.

TmParallelApply *An OS-independent parallel version of* `lapply`

Description

This function takes a vector or list and a function and applies in parallel.

Usage

```
TmParallelApply(
  X,
  FUN,
  cpus = parallel::detectCores(),
  export = NULL,
  libraries = NULL,
  envir = parent.frame()
)
```

Arguments

`X` A vector or list over which to apply `FUN`

`FUN` A function to apply over `X`

`cpus` Number of CPU cores to use, defaults to the value returned by `detectCores`.

`export` A character vector of objects in the workspace to export when using a Windows machine. Defaults to `NULL`

`libraries` A character vector of library/package names to load on to each cluster if using a Windows machine. Defaults to `NULL`

`envir` Environment from which to export variables in varlist

Details

This function is used to parallelize executions in `textmineR`. It is necessary because of differing capabilities between Windows and Unix. Unix systems use `mclapply`. Windows systems use `parLapply`.

Value

This function returns a `list` of length `length(X)`.

Examples

```
## Not run:
x <- 1:10000
f <- function(y) y * y + 12
result <- TmParallelApply(x, f)

## End(Not run)
```

update *Update methods for topic models*

Description

update will update a previously-trained topic model based on new data. Useful for updates or transfer learning.

Usage

```
update(object, ...)
```

Arguments

object	An existing trained topic model
...	Additional arguments to the call

update.lda_topic_model *Update a Latent Dirichlet Allocation topic model with new data*

Description

Update an LDA model with new data using collapsed Gibbs sampling.

Usage

```
## S3 method for class 'lda_topic_model'
update(
  object,
  dtm,
  additional_k = 0,
  iterations = NULL,
  burnin = -1,
  new_alpha = NULL,
  new_beta = NULL,
  optimize_alpha = FALSE,
  calc_likelihood = FALSE,
  calc_coherence = TRUE,
  calc_r2 = FALSE,
  ...
)
```

Arguments

`object` a fitted object of class `lda_topic_model`

`dtm` A document term matrix or term co-occurrence matrix of class dgCMatrix.

`additional_k` Integer number of topics to add, defaults to 0.

`iterations` Integer number of iterations for the Gibbs sampler to run. A future version may include automatic stopping criteria.

`burnin` Integer number of burnin iterations. If `burnin` is greater than -1, the resulting "phi" and "theta" matrices are an average over all iterations greater than `burnin`.

`new_alpha` For now not used. This is the prior for topics over documents used when updating the model

`new_beta` For now not used. This is the prior for words over topics used when updating the model.

`optimize_alpha` Logical. Do you want to optimize alpha every 10 Gibbs iterations? Defaults to `FALSE`.

`calc_likelihood` Do you want to calculate the likelihood every 10 Gibbs iterations? Useful for assessing convergence. Defaults to `FALSE`.

`calc_coherence` Do you want to calculate probabilistic coherence of topics after the model is trained? Defaults to `TRUE`.

`calc_r2` Do you want to calculate R-squared after the model is trained? Defaults to `FALSE`.

`...` Other arguments to be passed to `TmParallelApply`

Value

Returns an S3 object of class c("LDA", "TopicModel").

Examples

```
## Not run:
# load a document term matrix
d1 <- nih_sample_dtm[1:50,]

d2 <- nih_sample_dtm[51:100,]

# fit a model
m <- FitLdaModel(d1, k = 10,
                 iterations = 200, burnin = 175,
                 optimize_alpha = TRUE,
                 calc_likelihood = FALSE,
                 calc_coherence = TRUE,
                 calc_r2 = FALSE)

# update an existing model by adding documents
m2 <- update(object = m,
             dtm = rbind(d1, d2),
             iterations = 200,
             burnin = 175)

# use an old model as a prior for a new model
m3 <- update(object = m,
             dtm = d2, # new documents only
             iterations = 200,
             burnin = 175)

# add topics while updating a model by adding documents
m4 <- update(object = m,
             dtm = rbind(d1, d2),
             additional_k = 3,
             iterations = 200,
             burnin = 175)

# add topics to an existing model
m5 <- update(object = m,
             dtm = d1, # this is the old data
             additional_k = 3,
             iterations = 200,
             burnin = 175)


## End(Not run)
```

Index