Package 'textmineR'

October 6, 2016

Title Functions for Text Mining and Topic Modeling

Version 2.0.3

Date 2016-10-05

Author ``Thomas W. Jones <jones.thos.w@gmail.com>"

Maintainer Thomas W. Jones <jones.thos.w@gmail.com>

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 methods, lda, parallel, topicmodels, text2vec (>= 0.3), tm, stringr, SnowballC, Rcpp, RcppProgress, RSpectra

Suggests digest

License GPL (>= 3)

URL https://github.com/TommyJones/textmineR

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

LazyData true

LinkingTo Rcpp, RcppArmadillo, RcppProgress

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-10-06 19:29:16

R topics documented:

`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
```

`CalcPhiPrime` *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

```
CalcPhiPrime(phi, theta, p_docs = NULL)
```

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

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 phi_prime matrix, P(topic|words)
phi_prime <- GetPhiPrime(phi = nih_sample_topic_model$phi,
                         theta = nih_sample_topic_model$theta)
```

`CalcProbCoherence` *Probailistic coherence of topics*

Description

Calculates the probabilistic coherence of a topic or topics. This approximates semmantic 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-occurence matrix of class `matrix` or whose class inherits from the `Matrix` packge. 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 packge. 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)
```

`CorrectS` *Function to remove some forms of pluralization.*

Description

This function takes a character vector as input and removes some forms of pluralization from the ends of the words.

Usage

```
CorrectS(term_vec)
```

Arguments

`term_vec`	A character vector

Details

The entries of the vector should be single words or short n-grams without punctuation as the function only looks at the ends of strings. In other words, if entries are a paragraph of text. Only the final words will get de-pluralized. (Even then, if the final character is a period, as would be the case with paragraphs, it's likely that nothing will be de-pluralized.)

Value

Returns an object of class `data.frame` with three columns. The first column is the argument `term_vec`. The second column is the depluralized version of the words in `term_vec`. The third column is a logical, indicating whether or not the word in `term_vec` was changed.

Note

WARNING: This does make mistakes for irregular words. You should check its results manually. It tends to fail spectacularly for words ending in "es".

Examples

```
myvec <- c("banana", "bananas", "scientists", "large_armies")

CorrectS(term_vec=myvec)
```

`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(tm::stopwords("english"), tm::stopwords("SMART")),
  lower = TRUE, remove_punctuation = TRUE, remove_numbers = TRUE,
  stem_lemma_function = NULL, ...)
```

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(tm::stopwords("english"),` 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.
`...`	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-occurence matrix.*

Description

This is the main term co-occurence 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-occurence 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(tm::stopwords("english"), tm::stopwords("SMART")),
  lower = TRUE, remove_punctuation = TRUE, remove_numbers = TRUE,
  stem_lemma_function = NULL, ...)
```

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(tm::stopwords("english"), ... 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.

... — 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 times that term j appears across all documents containing i. Setting skipgram_window to 0 counts the number of documents in which term j and term i occur together. 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)
```

DepluralizeDtm *Run the CorrectS function on columns of a document term matrix.*

Description

Turns pluralizations of words in the columns of a document term matrix to their singular form. Then aggregates all columns that now have the same token. See example below.

Usage

```
DepluralizeDtm(dtm, ...)
```

Arguments

dtm	A document term matrix of class dgCMatrix whose colnames are tokens
...	Other arguments to pass to TmParallelApply. See note, below.

Value

Returns a document term matrix of class dgCMatrix. The columns index the de-pluralized tokens of the input document term matrix. In other words, there will generally be fewer columns in the returned matrix than the input matrix

Note

This function performs parallel computation by default. The default behavior is to use all available cores according to detectCores. However, this can be modified by passing the cpus argument when calling this function.

Examples

```
## Not run:
myvec <- c("the quick brown fox eats chickens",
           "the slow gray fox eats the slow chicken",
           "look at my horse", "my horses are amazing")

names(myvec) <- paste("doc", 1:length(myvec), sep="_")

dtm <- Vec2Dtm(vec = myvec, min.n.gram = 1, max.n.gram = 1)

dtm_new <- DepluralizeDtm(dtm = dtm)
#'
## 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 ]
```

Dtm2Tcm — *Turn a document term matrix into a term co-occurence matrix*

Description

Turn a document term matrix, whose rows index documents and whose columns index terms, into a term co-occurence matrix. A term co-occurence 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)
```

Files2Vec — *Function for reading text files into R*

Description

This function reads in all files in a directory ending in .txt into R. The result is a character vector where each entry is a .txt file. The names of the resulting vector are derived from the names of the files.

Usage

```
Files2Vec(directory, ...)
```

Arguments

`directory`	A path to directory containing the files you want to read into R.
`...`	Other arguments to be passed to `TmParallelApply`. See note, below.

Value

Returns a character vector where each entry corresponds to a document.

Note

This function performs parallel computation by default. The default behavior is to use all available cores according to `detectCores`. However, this can be modified by passing the `cpus` argument when calling this function.

Examples

```
## Not run:
my_text_vector <- Files2Vec("/path/to/my/data/")

## End(Not run)
```

`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, return_all = TRUE, ...)
```

Arguments

`dtm`	A document term matrix of class `dgCMatrix`
`k`	Number of topics
`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.

Value

Returns a list with a minumum 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.

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)
```

`FitLdaModel` *Fit a topic model using Latent Dirichlet Allocation*

Description

A wrapper for two implementations of Latent Dirichlet Allocation that returns a nicely-formatted topic model. See details, below.

Usage

```
FitLdaModel(dtm, k, iterations = NULL, alpha = 0.1, beta = 0.05,
  smooth = TRUE, method = "gibbs", return_all = FALSE, ...)
```

Arguments

`dtm`	A document term matrix of class `dgCMatrix`
`k`	Number of topics
`iterations`	The number of Gibbs iterations if `method = 'gibbs'`
`alpha`	Dirichlet parameter for the distribution of topics over documents. Defaults to 0.1
`beta`	Dirichlet parameter for the distribution of words over topics. Defaults to 0.05
`smooth`	Logical indicating whether or not you want to smooth the probabilities in the rows of `phi` and `theta`.
`method`	One of either 'gibbs' or 'vem' for either Gibbs sampling or variational expectation maximization. Defaults to 'gibbs'. See details, below.
`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 underlying functions. See details, below.

Details

For `method = 'gibbs'` this is a wrapper for `lda.collapsed.gibbs.sampler` from the `lda` package. Additional arguments can be passed to `lda.collapsed.gibbs.sampler` through `....` However, there are some arguments that, if passed through `...`, can cause conflicts. The arguments `K`, `alpha`, and `eta` for `lda.collapsed.gibbs.sampler` are set with the arguments `k`, `alpha`, and `beta`, respectively. The arguments `documents` and `vocab` for `lda.collapsed.gibbs.sampler` are set by `dtm` and aren't required.

For `method = 'vem'`, this function is a wrapper for `LDA` from the `topicmodels` library. Arguments to `LDA`'s `control` argument are passed through `....` `LDA`, by default, has behavior worth noting. By default, it estimates `alpha` and `beta` as part of the expectation maximization. Therefore, the values of `alpha` and `beta` passed to `LDA` will change unless `estimate.alpha` and `estimate.beta` are passed to `...` and set to `FALSE`.

The `...` argument can also be used to control the underlying behavior of `TmParallelApply`, such as the number of cpus, for example.

Value

Returns a list with a minumum 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.

Examples

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

# Fit an LDA model on a sample of documents
model <- FitLdaModel(dtm = nih_sample_dtm[ sample(1:nrow(nih_sample_dtm), 20), ],
                     k = 5, iterations = 200)

str(model)

# Fit a model, include likelihoods passed to lda::lda.collapsed.gibbs.sampler
model <- FitLdaModel(dtm = nih_sample_dtm[ sample(1:nrow(nih_sample_dtm), 20), ],
                     k = 5, iterations = 200, compute.log.likelihood = TRUE)

str(model)
```

`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, return_all = FALSE, ...)
```

Arguments

`dtm`	A document term matrix of class `Matrix::dgCMatrix`
`k`	Number of topics
`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)
```

FormatRawLdaOutput *Format Raw Output from* lda.collapsed.gibbs.sampler

---

Description

extracts outputs from LDA model estimated with `lda` package by Jonathan Chang

Usage

```
FormatRawLdaOutput(lda_result, docnames, smooth = TRUE)
```

Arguments

`lda_result`	The list value returned by `lda.collapsed.gibbs.sampler`
`docnames`	A character vector giving the names of documents. This is generally rownames(dtm).
`smooth`	Logical. Do you want to smooth your topic proportions so that there is a positive value for each term in each topic? Defaults to TRUE

Value

Returns a `list` with two elements: `phi` whose rows represent the distribution of words across a topic and `theta` whose rows represent the distribution of topics across a document.

Examples

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

# Get a sample of documents
dtm <- nih_sample_dtm[ sample(1:nrow(nih_sample_dtm), 20) , ]

# re-create a character vector of documents from the DTM
lex <- Dtm2Docs(dtm)

# Format for input to lda::lda.collapsed.gibbs.sampler
lex <- lda::lexicalize(lex, vocab=colnames(dtm))

# Fit the model from lda::lda.collapsed.gibbs.sampler
lda <- lda::lda.collapsed.gibbs.sampler(documents = lex, K = 5,
                                        vocab = colnames(dtm),
                                        num.iterations=200,
                                        alpha=0.1, eta=0.05)

# Format the result to get phi and theta matrices
lda <- FormatRawLdaOutput(lda_result=lda, docnames=rownames(dtm), smooth=TRUE)
```

GetPhiPrime *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. This function is deprecated. Use `CalcPhiPrime` instead.

Usage

```
GetPhiPrime(phi, theta)
```

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)

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 phi_prime matrix, P(topic|words)
phi_prime <- GetPhiPrime(phi = nih_sample_topic_model$phi,
                         theta = nih_sample_topic_model$theta)
```

`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 probabiliy 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)
```

Arguments

phi	A matrix whose rows index topics and columns index words
M	An integer for the number of terms to return

Value

Returns a `data.frame` 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_dtm)
data(nih_sample_topic_model)

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

str(top_terms)
```

HellDist *Hellinger Distance*

Description

Calculates the Hellinger distances or the rows or columns of a numeric matrix or for two numeric vectors. This function is deprecated. Use `CalcHellingerDist` instead.

Usage

```
HellDist(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
HellDist(x = x, y = y)

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

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.

JSD *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. This function is deprecated. Use CalcJSDivergence instead.

Usage

```
JSD(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
JSD(x = x, y = y)

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

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

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

assignments <- t(apply(nih_sample_topic_model$theta, 1, function(x){
  x[ x < 0.05 ] <- 0
```

```
  x / sum(x)
}))

labels <- LabelTopics(assignments = assignments, dtm = nih_sample_dtm, 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 http://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 http://exporter.nih.gov/ExPORTER_Catalog.aspx

---

`RecursiveRbind` *Recursively call rBind from the Matrix package.*

Description

This is used for combining a list of sparce matrices into a large Matrix. This is a worker function for `textmineR` and generally not made to be used by users. However, the function is provided for custom functions.

Usage

```
RecursiveRbind(matrix_list)
```

Arguments

matrix_list	A list, each element containing a matrix of class dgCMatrix

Value

Returns a matrix of class dgCMatrix.

Examples

```
a_matrix <- Matrix::Matrix(0, nrow=10, ncol=10)

a_list <- list(a_matrix, a_matrix, a_matrix, a_matrix, a_matrix)

result <- RecursiveRbind(a_list)
```

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 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)
```

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. Defauts 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)
```

Vec2Dtm *Convert a character vector to a document term matrix of class Matrix.*

Description

This function is deprecated. Use `CreateDtm` instead.

Usage

```
Vec2Dtm(vec, docnames = names(vec), min.n.gram = 1, max.n.gram = 1,
  remove.stopwords = TRUE, custom.stopwords = NULL, lower = TRUE,
  remove.punctuation = TRUE, remove.numbers = TRUE, stem.document = FALSE,
  ...)
```

Arguments

`vec`	A character vector of documents.
`docnames`	A vector of names for your documents. Defaults to `names(doc_vec)`. If NULL, then docnames is set to be `1:length(doc_vec)`.
`min.n.gram`	The minimum size of n for creating n-grams. Defaults to 1.
`max.n.gram`	The maximum size of n for creating n-grams. Defaults to 1. Numbers greater than 3 are discouraged due to risk of overfitting.
`remove.stopwords`	Do you want to remove standard stopwords from your documents? Defaults to `TRUE`.
`custom.stopwords`	If not `NULL` (the default) a character vector of stopwords to remove from your corpus.
`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.document`	Do you want to stem the words in your document using Porter's word stemmer? Defaults to `FALSE`
`...`	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.

Examples

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


dtm <- Vec2Dtm(vec = nih_sample$ABSTRACT_TEXT,
               docnames = nih_sample$APPLICATION_ID,
               min.n.gram = 1, max.n.gram = 2)

dim(dtm)

head(colnames(dtm))

head(rownames(dtm))

## End(Not run)
```

Index