Package ‘tosca’

April 20, 2021

Type Package

Title Tools for Statistical Content Analysis

Version 0.3-1

Date 2021-04-20

Description A framework for statistical analysis in content analysis. In addition to a pipeline for pre-processing text corpora and linking to the latent Dirichlet allocation from the 'lda' package, plots are offered for the descriptive analysis of text corpora and topic models. In addition, an implementation of Chang's intruder words and intruder topics is provided. Sample data for the vignette is included in the toscaData package, which is available on GitHub: <https://github.com/Docma-TU/toscaData>.

URL https://github.com/Docma-TU/tosca,
    https://doi.org/10.5281/zenodo.3591068

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports tm (>= 0.7-5), lda (>= 1.4.2), quanteda (>= 1.4.0), lubridate (>= 1.7.3), htmltools (>= 0.3.6), RColorBrewer (>= 1.1-2), stringr (>= 1.3.1), WikipediR (>= 1.5.0), data.table (>= 1.11.4)

Suggests toscaData, testthat (>= 2.0.0), knitr (>= 1.20), devtools (>= 1.13), rmarkdown (>= 1.9)

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Lars Koppers [aut, cre] (<https://orcid.org/0000-0002-1642-9616>), Jonas Rieger [aut] (<https://orcid.org/0000-0002-0007-4478>), Karin Boczek [ctb] (<https://orcid.org/0000-0003-1516-4094>), Gerret von Nordheim [ctb] (<https://orcid.org/0000-0001-7553-3838>)

Maintainer Lars Koppers <koppers@statistik.tu-dortmund.de>

Repository CRAN

Date/Publication 2021-04-20 09:50:02 UTC
R topics documented:

as.corpus.textmeta .................................................. 3
as.meta ................................................................. 4
as.textmeta.corpus .................................................... 5
cleanTexts .............................................................. 6
clusterTopics .......................................................... 7
deleteAndRenameDuplicates ......................................... 9
duplist .................................................................... 10
filterCount ............................................................... 12
filterDate ................................................................. 13
filterID .................................................................... 14
filterWord ............................................................... 15
intruderTopics ........................................................... 17
intruderWords ........................................................... 19
LDAgen ................................................................. 21
LDAprep ................................................................. 22
makeWordlist ............................................................ 23
mergeLDA ............................................................... 24
mergeTextmeta .......................................................... 25
plotArea ................................................................. 26
plotFreq ............................................................... 27
plotHeat ............................................................... 30
plotScot ............................................................. 31
plotTopic ............................................................. 33
plotTopicWord .......................................................... 35
plotWordpt ............................................................ 38
plotWordSub ............................................................ 40
precision ............................................................... 43
readTextmeta .......................................................... 44
readWhatsApp .......................................................... 45
readWiki .............................................................. 46
readWikinews ........................................................... 46
removeXML ............................................................ 47
sampling ............................................................... 48
showMeta ............................................................. 49
showTexts ............................................................. 50
textmeta ............................................................... 51
tidy.textmeta .......................................................... 52
topicCoherence ........................................................ 53
topicsInText ........................................................... 54
topTexts ............................................................... 56
topWords .............................................................. 57
Index .................................................................. 59
as.corpus.textmeta  Transform textmeta to corpus

Description

Transfers data from a textmeta object to a corpus object - the way text data is stored in the package quanteda.

Usage

as.corpus.textmeta(
  object,
  docnames = "id",
  docvars = setdiff(colnames(object$meta), "id"),
  ...
)

Arguments

object textmeta object
docnames Character: string with the column of object$meta which should be kept as docnames.
docvars Character: vector with columns of object$meta which should be kept as docvars.
... Additional parameters like meta or compress for corpus.

Value

corpus object

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals."
)

obj <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

corp <- as.corpus.textmeta(obj)
quanteda::docvars(corp)
#quanteda::textstat_summary(corp)
Description

Helper to create the requested data.frame to create a "textmeta" object.

Usage

```r
as.meta(
  x,
  cols = colnames(x),
  idCol = "id",
  dateCol = "date",
  titleCol = "title",
  dateFormat
)
```

Arguments

- `x` data.frame to convert
- `cols` character vector with columns which should be kept
- `idCol` character string with column name of the IDs
- `dateCol` character string with column name of the Dates
- `titleCol` character string with column name of the Titles
- `dateFormat` character string with the date format in x for `as.Date`. If not supplied, dates are not transformed.

Value

A data.frame with columns "id", "date", "title" and user-specified others.

Examples

```r
meta <- data.frame(id = 1:3, additionalVariable = matrix(5, ncol = 4, nrow = 3))
(as.meta(meta))
```
as.textmeta.corpus  Transform corpus to textmeta

Description

Transfers data from a corpus object - the way text data is stored in the package quanteda - to a textmeta object.

Usage

as.textmeta.corpus(
corpus,
cols,
dateFormat = "%Y-%m-%d",
idCol = "id",
dateCol = "date",
titleCol = "title",
textCol = "texts",
duplicateAction = TRUE,
addMetadata = TRUE
)

Arguments

corpus  Object of class corpus, package quanteda.
cols  Character: vector with columns which should be kept.
dateFormat  Character: string with the date format in the date column for as.Date.
idCol  Character: string with column name of the IDs in corpus - named "id" in the resulting data.frame.
dateCol  Character: string with column name of the Dates in corpus - named "date" in the resulting data.frame.
titleCol  Character: string with column name of the Titles in corpus - named "title" in the resulting data.frame.
textCol  Character: string with column name of the Texts in corpus - results in a named list ("id") of the Texts.
duplicateAction  Logical: Should deleteAndRenameDuplicates be applied to the created textmeta object?
addMetadata  Logical: Should the metadata flag of corpus be added to the meta flag of the textmeta object? If there are conflicts regarding the naming of columns, the metadata columns would be overwritten by the document specific columns.

Value

textmeta object
Examples

texts <- c("Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
"So Long, and Thanks for All the Fish",
"A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

corp <- quanteda::corpus(x = texts)
obj <- as.textmeta.corpus(corp, addMetadata = FALSE)

quanteda::docvars(corp, "title") <- c("Fishing", "Don't panic!", "Sir Ronald")
quanteda::docvars(corp, "date") <- c("1885-01-02", "1979-03-04", "1951-05-06")
quanteda::docvars(corp, "id") <- c("A", "B", "C")
quanteda::docvars(corp, "additionalVariable") <- 1:3

obj <- as.textmeta.corpus(corp)

cleanTexts  
Data Preprocessing

Description

Removes punctuation, numbers and stopwords, changes letters into lowercase and tokenizes.

Usage

cleanTexts(
  object,  
text,  
sw = "en",  
paragraph = FALSE,  
lowercase = TRUE,  
rmPunctuation = TRUE,  
rmNumbers = TRUE,  
checkUTF8 = TRUE,  
ucp = TRUE  
)

Arguments

object  
textmeta object

text  
Not necassary if object is specified, else should be object$text: List of article texts.

sw  
Character: Vector of stopwords. If the vector is of length one, sw is interpreted as argument for stopwords from the tm package.

paragraph  
Logical: Should be set to TRUE if one article is a list of character strings, representing the paragraphs.
lowercase Logical: Should be set to TRUE if all letters should be coerced to lowercase.
rmpPunctuation Logical: Should be set to TRUE if punctuation should be removed from articles.
rmNumbers Logical: Should be set to TRUE if numbers should be removed from articles.
checkUTF8 Logical: Should be set to TRUE if articles should be tested on UTF-8 - which is package standard.
ucp Logical: ucp option for removePunctuation from the tm package. Runs remove punctuation twice (ASCII and Unicode).

Details

Removes punctuation, numbers and stopwords, change into lowercase letters and tokenization. Additional some cleaning steps: remove empty words / paragraphs / article.

Value

A textmeta object or a list (if object is not specified) containing the preprocessed articles.

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")
corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)
cleanTexts(object=corpus)
texts <- list(A=c("Give a Man a Fish, and You Feed Him for a Day.",
"Teach a Man To Fish, and You Feed Him for a Lifetime"),
B="So Long, and Thanks for All the Fish",
C=c("A very able manipulative mathematician.",
"Fisher enjoys a real mastery in evaluating complicated multiple integrals."))
cleanTexts(text=texts, sw = "en", paragraph = TRUE)

clusterTopics

Cluster Analysis

Description

This function makes a cluster analysis using the Hellinger distance.
Usage

clusterTopics(
  ldaresult,
  file,
  tnames = NULL,
  method = "average",
  width = 30,
  height = 15,
  ...
)

Arguments

ldaresult The result of a function call LDAgen - alternatively the corresponding matrix
result$topics

file File for the dendogram pdf.

tnames Character vector as label for the topics.

method Method statement from hclust

width Grafical parameter for pdf output. See pdf

height Grafical parameter for pdf output. See pdf

... Additional parameter for plot

Details

This function is useful to analyze topic similarities and while evaluating the right number of topics
of LDAs.

Value

A dendogram as pdf and a list containing

dist A distance matrix

clust The result from hclust

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery
in evaluating complicated multiple integrals.")
corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)
corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
ldaPrep <- LDAprep(text=corpus$text, vocab=wordlist$words)

LDA <- LDAgen(documents=ldaPrep, K = 3L, vocab=wordlist$words, num.words=3)
clusterTopics(ldaresult=LDA)

deleteAndRenameDuplicates

*Deletes and Renames Articles with the same ID*

**Description**

Deletes articles with the same ID and same text. Renames the ID of articles with the same ID but different text-component (_IDFakeDup, _IDRealDup).

**Usage**

deleteAndRenameDuplicates(object, renameRemaining = TRUE)

**Arguments**

- `object`: A `textmeta` object as a result of a read-function.
- `renameRemaining`: Logical: Should all articles for which a counterpart with the same id exists, but which do not have the same text and - in addition - which matches (an)other article(s) in the text field be named a "fake duplicate" or not.

**Details**

Summary: Different types of duplicates: "complete duplicates" = same ID, same information in text, same information in meta "real duplicates" = same ID, same information in text, different information in meta "fake duplicates" = same ID, different information in text

**Value**

A filtered `textmeta` object with updated IDs.

**Examples**

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
A="A fake duplicate",
B="So Long, and Thanks for All the Fish",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.",
C="A very able manipulative mathematician, Fisher enjoys a real mastery
in evaluating complicated multiple integrals.

```r
corpus <- textmeta(meta=data.frame(id=c("A", "A", "B", "B", "C", "C"),
   title=c("Fishing", "Fake duplicate", "Don't panic!", "towel day", "Sir Ronald", "Sir Ronald"),
   date=c("1885-01-02", "1885-01-03", "1979-03-04", "1979-03-05", "1951-05-06", "1951-05-06"),
   stringsAsFactors=FALSE), text=texts)
duplicates <- deleteAndRenameDuplicates(object=corpus)
duplicates$meta$id

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
   Teach a Man To Fish, and You Feed Him for a Lifetime",
   A="Give a Man a Fish, and You Feed Him for a Day.
   Teach a Man To Fish, and You Feed Him for a Lifetime",
   A="A fake duplicate",
   B="So Long, and Thanks for All the Fish",
   B="So Long, and Thanks for All the Fish",
   C="A very able manipulative mathematician, Fisher enjoys a real mastery
   in evaluating complicated multiple integrals.
   A very able manipulative mathematician, Fisher enjoys a real mastery
   in evaluating complicated multiple integrals.")
corpus <- textmeta(meta=data.frame(id=c("A", "A", "A", "B", "B", "C", "C"),
   title=c("Fishing", "Fishing2", "Fake duplicate", "Don't panic!", "towel day", "Sir Ronald", "Sir Ronald"),
   date=c("1885-01-02", "1885-01-02", "1885-01-03", "1979-03-04", "1979-03-05", "1951-05-06", "1951-05-06"),
   stringsAsFactors=FALSE), text=texts)
duplicates <- deleteAndRenameDuplicates(object=corpus)
duplicates2 <- deleteAndRenameDuplicates(object=corpus, renameRemaining = FALSE)
```

---

duplist  

Creating List of Duplicates

**Description**

Creates a List of different types of Duplicates in a textmeta-object.

**Usage**

```r
duplist(object, paragraph = FALSE)
is.duplist(x)
## S3 method for class 'duplist'
print(x, ...)
## S3 method for class 'duplist'
summary(object, ...)
```
Arguments

object A textmeta-object.
paragraph Logical: Should be set to TRUE if the article is a list of character strings, representing the paragraphs.
x An R Object.
... Further arguments for print and summary. Not implemented.

Details

This function helps to identify different types of Duplicates and gives the ability to exclude these for further Analysis (e.g. LDA).

Value

Named List:

uniqueTexts Character vector of IDs so that each text occurs once - if a text occurs twice or more often in the corpus, the ID of the first text regarding the list-order is returned
notDuplicatedTexts Character vector of IDs of texts which are represented only once in the whole corpus
idFakeDups List of character vectors: IDs of texts which originally has the same ID but belongs to different texts grouped by their original ID
idRealDups List of character vectors: IDs of texts which originally has the same ID and text but different meta information grouped by their original ID
allTextDups List of character vectors: IDs of texts which occur twice or more often grouped by text equality
textMetaDups List of character vectors: IDs of texts which occur twice or more often and have the same meta information grouped by text and meta equality

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime", A="A fake duplicate", B="So Long, and Thanks for All the Fish", B="So Long, and Thanks for All the Fish", C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals."). C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")
filterCount <- deleteAndRenameDuplicates(object=corpus)
duplist(object=duplicates, paragraph = FALSE)

---

**filterCount**  
*Subcorpus With Count Filter*

**Description**

Generates a subcorpus by restricting it to texts containing a specific number of words.

**Usage**

```
filterCount(...)
```

```r
## Default S3 method:
filterCount(text, count = 1L, out = c("text", "bin", "count"), ...)
```

```r
## S3 method for class 'textmeta'
filterCount(
  object,  
  count = 1L,  
  out = c("text", "bin", "count"),  
  filtermeta = TRUE,  
  ...
)
```

**Arguments**

- `...`: Not used.
- `text`: Not necessary if `object` is specified, else should be `object$text`: list of article texts.
- `count`: An integer marking how many words must at least be found in the text.
- `out`: Type of output: `text` filtered corpus, `bin` logical vector for all texts, `count` the counts.
- `object`: A `textmeta` object.
- `filtermeta`: Logical: Should the meta component be filtered, too?

**Value**

A `textmeta` object if `object` is specified, else only the filtered text. If a `textmeta` object is returned its meta data are filtered to those texts which appear in the corpus by default (`filtermeta`).
Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
              Teach a Man To Fish, and You Feed Him for a Lifetime",
              B="So Long, and Thanks for All the Fish",
              C="A very able manipulative mathematician, Fisher enjoys a real mastery
              in evaluating complicated multiple integrals.")

filterCount(text=texts, count=10L)
filterCount(text=texts, count=10L, out="bin")
filterCount(text=texts, count=10L, out="count")

---

filterDate

Subcorpus With Date Filter

Description

Generates a subcorpus by restricting it to a specific time window.

Usage

filterDate(...)

## Default S3 method:
filterDate(
  text,
  meta,
  s.date = min(meta$date, na.rm = TRUE),
  e.date = max(meta$date, na.rm = TRUE),
  ...
)

## S3 method for class 'textmeta'
filterDate(
  object,
  s.date = min(object$meta$date, na.rm = TRUE),
  e.date = max(object$meta$date, na.rm = TRUE),
  filtermeta = TRUE,
  ...
)

Arguments

... Not used.
text Not necessary if object is specified, else should be object$text
meta Not necessary if object is specified, else should be object$meta
s.date Start date of subcorpus as date object
filterID

Subcorpus With ID Filter

Description

Generates a subcorpus by restricting it to specific ids.

Usage

filterID(...)  

## Default S3 method:
filterID(text, id, ...)

## S3 method for class 'textmeta'
filterID(object, id, filtermeta = TRUE, ...)

dates  
edate End date of subcorpus as date object  
object textmeta object  
filtermeta Logical: Should the meta component be filtered, too?

Value

textmeta object if object is specified, else only the filtered text. If a textmeta object is returned its meta data are filtered to those texts which appear in the corpus by default (filtermeta).

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

subcorpus <- filterDate(object=corpus, s.date = "1951-05-06")
subcorpus$meta
subcorpus$text
filterWord

Subcorpus With Word Filter

Description
Generates a subcorpus by restricting it to texts containing specific filter words.

Usage

filterWord(...)

## Default S3 method:
filterWord(
  text,
  search,
  ignore.case = FALSE,
)
## S3 method for class 'textmeta'
filterWord(
  object,
  search,
  ignore.case = FALSE,
  out = c("text", "bin", "count"),
  filtermeta = TRUE,
  ...
)

### Arguments

- **text**: Not used.
- **search**: List of data frames. Every List element is an 'or' link, every entry in a data frame is linked by an 'and'. The dataframe must have following tree variables: pattern a character string including the search terms, word, a logical value displaying if a word (TRUE) or character (search) is wanted and count an integer marking how many times the word must at least be found in the text. word can alternatively be a character string containing the keywords pattern for character search, word for word-search and left and right for truncated search. If search is only a character Vector the link is 'or', and a character search will be used with count=1
- **ignore.case**: Logical: Lower and upper case will be ignored.
- **out**: Type of output: text filtered corpus, bin logical vector for all texts, count the number of matches.
- **object**: A textmeta object
- **filtermeta**: Logical: Should the meta component be filtered, too?

### Value

- textmeta object if object is specified, else only the filtered text. If a textmeta object is returned its meta data are filtered to those texts which appear in the corpus by default (filtermeta).

### Examples

```r
texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

# search for pattern "fish"
```
intruderTopics

Function to validate the fit of the LDA model

Description

This function validates a LDA result by presenting a mix of topics and intruder topics to a human user, who has to identify them.

Usage

intruderTopics(
  text = NULL,
  beta = NULL,
  theta = NULL,
  id = NULL,
  numIntruder = 1,
  numOuttopics = 4,
  byScore = TRUE,
  minWords = 0L,
  minOuttopics = 0L,
  stopTopics = NULL,
  printSolution = FALSE,
  oldResult = NULL,
  test = FALSE,
  testinput = NULL
)
intruderTopics

Arguments

- **text**: A list of texts (e.g. the text element of a `textmeta` object).
- **beta**: A matrix of word-probabilities or frequency table for the topics (e.g. the topics matrix from the `LDAGen` result). Each row is a topic, each column a word. The rows will be divided by the row sums, if they are not 1.
- **theta**: A matrix of word counts per text and topic (e.g. the document_sums matrix from the `LDAGen` result). Each row is a topic, each column a text. In each cell stands the number of words in text j belonging to topic i.
- **id**: Optional: character vector of text IDs that should be used for the function. Useful to start an inchoate coding task.
- **numIntruder**: Intended number of intruder words. If numIntruder is a integer vector, the number would be sampled for each topic.
- **numOuttopics**: tba Integer: Number of words per topic, including the intruder words
- **byScore**: Logical: Should the score of top_topic.words from the lda package be used?
- **minWords**: Integer: Minimum number of words for a choosen text.
- **minOuttopics**: Integer: Minimal number of words a topic needs to be classified as a possible correct Topic.
- **stopTopics**: Optional: Integer vector to deselect stopword topics for the coding task.
- **printSolution**: Logical: If TRUE the coder gets a feedback after his/her vote.
- **oldResult**: Result object from an unfinished run of `intruderWords`. If oldResult is used, all other parameter will be ignored.
- **test**: Logical: Enables test mode
- **testinput**: Input for function tests

Value

Object of class IntruderTopics. List of 11

- **result**: Matrix of 3 columns. Each row represents one labeled text. numIntruder (1. column) gives the number of intruder topics inputated in this text, missIntruder (2. column) the number of the intruder topics which were not found by the coder and falseIntruder (3. column) the number of the topics choosen by the coder which were no intruder.
- **beta**: Parameter of the function call
- **theta**: Parameter of the function call
- **id**: Character Vector of IDs at the beginning
- **byScore**: Parameter of the function call
- **numIntruder**: Parameter of the function call
- **numOuttopics**: Parameter of the function call
- **minWords**: Parameter of the function call
- **minOuttopics**: Parameter of the function call
- **unusedID**: Character vector of unused text IDs for the next run
- **stopTopics**: Parameter of the function call
intruderWords

References


Examples

## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDAprep(text=poliClean$text, vocab=words10)
LDAresult <- LDAgen(documents=poliLDA, K=10, vocab=words10)
intruder <- intruderTopics(text=politics$text, beta=LDAresult$topics,
                          theta=LDAresult$document_sums, id=names(poliLDA))

## End(Not run)

intruderWords  

Function to validate the fit of the LDA model

Description

This function validates a LDA result by presenting a mix of words from a topic and intruder words to a human user, who has to identity them.

Usage

intruderWords(
  beta = NULL,
  byScore = TRUE,
  numTopwords = 30L,
  numIntruder = 1L,
  numOutwords = 5L,
  noTopic = TRUE,
  printSolution = FALSE,
  oldResult = NULL,
  test = FALSE,
  testinput = NULL
)

Arguments

beta  A matrix of word-probabilities or frequency table for the topics (e.g. the topics matrix from the LDAgen result). Each row is a topic, each column a word. The rows will be divided by the row sums, if they are not 1.

byScore  Logical: Should the score of top.topic.words from the lda package be used?
intruderWords

numTopwords  The number of topwords to be used for the intruder words
numIntruder  Intended number of intruder words. If numIntruder is a integer vector, the
             number would be sampled for each topic.
numOutwords  Integer: Number of words per topic, including the intruder words.
noTopic      Logical: Is x input allowed to mark nonsense topics?
printSolution tba
oldResult    Result object from an unfinished run of intruderWords. If oldResult is used,
             all other parameter will be ignored.
test         Logical: Enables test mode
testinput    Input for function tests

Value

Object of class IntruderWords. List of 7

result       Matrix of 3 columns. Each row represents one topic. All values are 0 if the
             topic did not run before. numIntruder (1. column) gives the number of in-
             truder words inputated in this topic, missIntruder (2. column) the number of
             the intruder words which were not found by the coder and falseIntruder (3.
             column) the number of the words choosen by the coder which were no intruder.

beta         Parameter of the function call
byScore      Parameter of the function call
numTopwords  Parameter of the function call
numIntruder  Parameter of the function call
numOutwords  Parameter of the function call
noTopic      Parameter of the function call

References

Chang, Jonathan and Sean Gerrish and Wang, Chong and Jordan L. Boyd-graber and David M.

Examples

## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDApref(text=poliClean$text, vocab=words10)
LDAreresult <- LDAgen(documents=poliLDA, K=10, vocab=words10)
intruder <- intruderWords(beta=LDAreresult$topics)
## End(Not run)
LDAnegn

Function to fit LDA model

Description

This function uses the \texttt{lda.collapsed.gibbs.sampler} from the \texttt{lda-} package and additionally saves topword lists and a \texttt{R} workspace.

Usage

\begin{verbatim}
LDAnegn(
    documents,
    K = 100L,
    vocab,
    num.iterations = 200L,
    burnin = 70L,
    alpha = NULL,
    eta = NULL,
    seed = NULL,
    folder = file.path(tempdir(), "lda-result"),
    num.words = 50L,
    LDA = TRUE,
    count = FALSE
)
\end{verbatim}

Arguments

- \texttt{documents} A list prepared by \texttt{LDAnep}.  
- \texttt{K} Number of topics  
- \texttt{vocab} Character vector containing the words in the corpus  
- \texttt{num.iterations} Number of iterations for the gibbs sampler  
- \texttt{burnin} Number of iterations for the burnin  
- \texttt{alpha} Hyperparameter for the topic proportions  
- \texttt{eta} Hyperparameter for the word distributions  
- \texttt{seed} A seed for reproducability.  
- \texttt{folder} File for the results. Saves in the temporary directionary by default.  
- \texttt{num.words} Number of words in the top topic words list  
- \texttt{LDA} logical: Should a new model be fitted or an existing \texttt{R} workspace?  
- \texttt{count} logical: Should article counts calculated per top topic words be used for output as csv (default: FALSE)?

Value

A .csv file containing the topword list and a \texttt{R} workspace containing the result data.
References


See Also

Documentation for the lda package.

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
LDAprep <- LDAprep(text=corpus$text, vocab=wordlist$words)

LDAgen(documents=LDAprep, K = 3L, vocab=wordlist$words, num.words=3)

LDAprep

Create Lda-ready Dataset

Description

This function transforms a text corpus such as the result of cleanTexts into the form needed by the lda-package.

Usage

LDAprep(text, vocab, reduce = TRUE)

Arguments

text A list of tokenized texts
vocab A character vector containing all words which should be used for lda
reduce Logical: Should empty texts be deleted?
Value

A list in which every entry contains a matrix with two rows: The first row gives the number of the entry of the word in vocab minus one, the second row is 1 and the number of the occurrence of the word will be shown by the number of columns belonging to this word.

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime", B="So Long, and Thanks for All the Fish", C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"), title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"), date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"), additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)
corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
LDAprep(text=corpus$text, vocab=wordlist$words, reduce = TRUE)

makeWordlist

Counts Words in Text Corpora

Description

Creates a wordlist and a frequency table.

Usage

makeWordlist(text, k = 100000L, ...)

Arguments

text
  List of texts.
k
  Integer: How many texts should be processed at once (RAM usage)?
... further arguments for the sort function. Often you want to set method = "radix".

Details

This function helps, if table(x) needs too much RAM.

Value

words An alphabetical list of the words in the corpus
wordtable A frequency table of the words in the corpus
mergeLDA

Preparation of Different LDAs For Clustering

Description

Merges different lda-results to one matrix, including only the words which appears in all lda-results.

Usage

mergeLDA(x)

Arguments

x

A list of lda results.

Details

The function is useful for merging lda-results prior to a cluster analysis with clusterTopics.

Value

A matrix including all topics from all lda-results. The number of rows is the number of topics, the number of columns is the number of words which appear in all results.

Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime", B="So Long, and Thanks for All the Fish", C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

texts <- cleanTexts(text=texts)
makeWordlist(text=texts, k = 2L)

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"), title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"), date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"), additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
ldaPrep <- LDAprep(text=corpus$text, vocab=wordlist$words)

LDA1 <- LDAGen(documents=ldaPrep, K = 3L, vocab=wordlist$words, num.words=3)
LDA2 <- LDAGen(documents=ldaPrep, K = 3L, vocab=wordlist$words, num.words=3)
mergeLDA(list(LDA1=LDA1, LDA2=LDA2))

### mergeTextmeta

#### Merge Textmeta Objects

**Description**

Merges a list of textmeta objects to a single object. It is possible to control whether all columns or the intersect should be considered.

**Usage**

mergeTextmeta(x, all = TRUE)

**Arguments**

- **x**: A list of textmeta objects
- **all**: Logical: Should the result contain union (TRUE) or intersection (FALSE) of columns of all objects? If TRUE, the columns which at least appear in one of the meta components are filled with NAs in the merged meta component.

**Value**

textmeta object

**Examples**

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery
in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

corpus2 <- textmeta(meta=data.frame(id=c("E", "F"),
title=c("title1", "title2"),
date=c("2018-01-01", "2018-01-01"),
additionalVariable2=1:2, stringsAsFactors=FALSE), text=list(E="text1", F="text2"))

merged <- mergeTextmeta(x=list(corpus, corpus2), all = TRUE)
str(merged$meta)

merged <- mergeTextmeta(x=list(corpus, corpus2), all = FALSE)
str(merged$meta)

---

**plotArea**

*Plotting topics over time as stacked areas below plotted lines.*

**Description**

Creates a stacked area plot of all or selected topics.

**Usage**

```r
plotArea(
  ldaresult,
  ldaID,
  select = NULL,
  tnames = NULL,
  threshold = NULL,
  meta,
  unit = "quarter",
  xunit = "year",
  color = NULL,
  sort = TRUE,
  legend = NULL,
  legendLimit = 0,
  peak = 0,
  file
)
```

**Arguments**

- `ldaresult` LDA result object
- `ldaID` Character vector including IDs of the texts
- `select` Selects all topics if parameter is null. Otherwise vector of integers or topic label. Only topics belonging to that numbers, and labels respectively would be plotted.
- `tnames` Character vector of topic labels. It must have same length than number of topics in the model.
- `threshold` Numeric: Treshold between 0 and 1. Topics would only be used if at least one time unit exist with a topic proportion above the treshold
- `meta` The meta data for the texts or a date-string.
- `unit` Time unit for x-axis. Possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see `round_date`
- `xunit` Time unit for tiks on the x-axis. For possible units see `round_date`
plotFreq

plotFreq is useful to visualize the volume of topics and to show trends over time.

Details

This function is useful to visualize the volume of topics and to show trends over time.

Value

List of two matrices. rel contains the topic proportions over time, relcum contains the cumulated topic proportions

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
polilDA <- LDAprep(text=poliClean$text, vocab=words10)
LDArresult <- LDAgen(documents=polilDA, K=10, vocab=words10)
plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta)

plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta, select=c(1,3,5))

## End(Not run)
```

Description

plotFreq is useful to visualize the volume of topics and to show trends over time.

This function is useful to visualize the volume of topics and to show trends over time.

Value

List of two matrices. rel contains the topic proportions over time, relcum contains the cumulated topic proportions

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
polilDA <- LDAprep(text=poliClean$text, vocab=words10)
LDArresult <- LDAgen(documents=polilDA, K=10, vocab=words10)
plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta)

plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta, select=c(1,3,5))

## End(Not run)
```

plotFreq

Plotting Counts of specified Wordgroups over Time (relative to Corpus)

Description

plotFreq is useful to visualize the volume of topics and to show trends over time.

This function is useful to visualize the volume of topics and to show trends over time.

Value

List of two matrices. rel contains the topic proportions over time, relcum contains the cumulated topic proportions

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
polilDA <- LDAprep(text=poliClean$text, vocab=words10)
LDArresult <- LDAgen(documents=polilDA, K=10, vocab=words10)
plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta)

plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta, select=c(1,3,5))

## End(Not run)
```

Description

plotFreq is useful to visualize the volume of topics and to show trends over time.

This function is useful to visualize the volume of topics and to show trends over time.

Value

List of two matrices. rel contains the topic proportions over time, relcum contains the cumulated topic proportions

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
polilDA <- LDAprep(text=poliClean$text, vocab=words10)
LDArresult <- LDAgen(documents=polilDA, K=10, vocab=words10)
plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta)

plotArea(ldaresult=LDArresult, ldaID=names(polilDA), meta=politics$meta, select=c(1,3,5))

## End(Not run)
```
Usage

plotFreq(
  object,
  id = names(object$text),
  type = c("docs", "words"),
  wordlist,
  link = c("and", "or"),
  wnames,
  ignore.case = FALSE,
  rel = FALSE,
  mark = TRUE,
  unit = "month",
  curves = c("exact", "smooth", "both"),
  smooth = 0.05,
  both.lwd,
  both.lty,
  main,
  xlab,
  ylab,
  ylim,
  col,
  legend = "topright",
  natozero = TRUE,
  file,
  ...
)

Arguments

object  textmeta object with strictly tokenized text component (character vectors) -
        like a result of cleanTexts
id      character vector (default: object$meta$id) which IDs specify the subcorpus
type    character (default: "docs") should counts/proportion of documents, where
        every "docs" or words "words" be plotted
wordlist list of character vectors. Every list element is an 'or' link, every character
        string in a vector is linked by the argument link. If wordlist is only a character
        vector it will be coerced to a list of the same length as the vector (see as.list),
        so that the argument link has no effect. Each character vector as a list element
        represents one curve in the outcoming plot
link    character (default: "and") should the (inner) character vectors of each list
        element be linked by an "and" or an "or"
wnames  character vector of same length as wordlist - labels for every group of 'and'
        linked words
ignore.case logical (default: FALSE) option from grepl.
rel      logical (default: FALSE) should counts (FALSE) or proportion (TRUE) be plotted
mark     logical (default: TRUE) should years be marked by vertical lines
unit character (default: "month") to which unit should dates be floored. Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see `round_date`.

curves character (default: "exact") should "exact", "smooth" curve or "both" be plotted.

smooth numeric (default: 0.05) smoothing parameter which is handed over to `lowess` as f

both.lwd graphical parameter for smoothed values if curves = "both"

both.lty graphical parameter for smoothed values if curves = "both"

main character graphical parameter

xlab character graphical parameter

ylab character graphical parameter

ylim (default if rel = TRUE: c(0,1)) graphical parameter

col graphical parameter, could be a vector. If curves = "both" the function will for every wordgroup plot at first the exact and then the smoothed curve - this is important for your col order.

legend character (default: "topright") value(s) to specify the legend coordinates. If "none" no legend is plotted.

natozero logical (default: TRUE) should NAs be coerced to zeros. Only has effect if rel = TRUE.

file character file path if a pdf should be created

... additional graphical parameters

Value

A plot. Invisible: A dataframe with columns date and wnames - and additionally columns wnames_rel for rel = TRUE - with the counts (and proportion) of the given wordgroups.

Examples

```r
# Not run:
data(politics)
poliClean <- cleanTexts(politics)
plotFreq(poliClean, wordlist=c("obama", "bush"))
```

# End(Not run)
plotHeat

Plotting Topics over Time relative to Corpus

Description

Creates a pdf showing a heat map. For each topic, the heat map shows the deviation of its current share from its mean share. Shares can be calculated on corpus level or on subcorpus level concerning LDA vocabulary. Shares can be calculated in absolute deviation from the mean or relative to the mean of the topic to account for different topic strengths.

Usage

```r
plotHeat(
  object,
  ldaresult,
  ldaID,
  select = 1:nrow(ldaresult$document_sums),
  tnames,
  norm = FALSE,
  file,
  unit = "year",
  date_breaks = 1,
  margins = c(5, 0),
  ...
)
```

Arguments

- `object`: `textmeta` object with strictly tokenized text component (calculation of proportion on document lengths) or `textmeta` object which contains only the meta component (calculation of proportion on count of words out of the LDA vocabulary in each document)
- `ldaresult`: LDA result object.
- `ldaID`: Character vector containing IDs of the texts.
- `select`: Numeric vector containing the numbers of the topics to be plotted. Defaults to all topics.
- `tnames`: Character vector with labels for the topics.
- `norm`: Logical: Should the values be normalized by the mean topic share to account for differently sized topics (default: `FALSE`)?
- `file`: Character vector containing the path and name for the pdf output file.
- `unit`: Character: To which unit should dates be floored (default: "year")? Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see `round_date`.
- `date_breaks`: How many labels should be shown on the x axis (default: 1)? If `date_breaks` is 5 every fifth label is drawn.
margins
... See heatmap

Additional graphical parameters passed to heatmap, for example distfun or hclustfun. The function is useful to search for peaks in the coverage of topics.

Value


Examples

## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDAprep(text=poliClean$text, vocab=words10)
LDAresult <- LDAgen(documents=poliLDA, K=10, vocab=words10)
plotHeat(object=poliClean, ldaresult=LDAresult, ldaID=names(poliLDA))

## End(Not run)
natozero = TRUE,  
file,  
...  
)

Arguments

object  textmeta object with strictly tokenized text component vectors if type = "words"  

id  Character: Vector (default: object$meta$id) which IDs specify the subcorpus  

type  Character: Should counts/proportion of documents "docs" (default) or words "words" be plotted?  

rel  Logical: Should counts (default: FALSE) or proportion (TRUE) be plotted?  

mark  Logical: Should years be marked by vertical lines (default: TRUE)?  

unit  Character: To which unit should dates be floored (default: "month"). Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see round_date.  

curves  Character: Should "exact", "smooth" curve or "both" be plotted (default: "exact")?  

smooth  Numeric: Smoothing parameter which is handed over to lowess as f (default: 0.05).  

main  Character: Graphical parameter  

xlab  Character: Graphical parameter  

ylab  Character: Graphical parameter  

ylim  Graphical parameter (default if rel = TRUE: c(0,1))  

both.lwd  Graphical parameter for smoothed values if curves = "both"  

both.col  Graphical parameter for smoothed values if curves = "both"  

both.lty  Graphical parameter for smoothed values if curves = "both"  

natozero  Logical: Should NAs be coerced to zeros (default: TRUE)? Only has an effect if rel = TRUE.  

file  Character: File path if a pdf should be created.  

...  additional graphical parameters

Details

object needs a textmeta object with strictly tokenized text component (character vectors) if you use type = "words". If you use type = "docs" you can use a tokenized or a non-tokenized text component. In fact, you can use the textmeta constructor (textmeta(meta = <your-meta-data.frame>)) to create a textmeta object containing only the meta field and plot the resulting object. This way you can save time and memory at the first glance.

Value

A plot Invisible: A dataframe with columns date and counts, respectively proportion
Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)

# complete corpus
plotScot(object=poliClean)

# subcorpus
subID <- filterWord(poliClean, search=c("bush", "obama"), out="bin")
plotScot(object=poliClean, id=names(subID)[subID], curves="both", smooth=0.3)
## End(Not run)
```

plotTopic  
Plotting Counts of Topics over Time (Relative to Corpus)

Description

Creates a plot of the counts/proportion of specified topics of a result of `LDAgen`. There is an option to plot all curves in one plot or to create one plot for every curve (see `pages`). In addition the plots can be written to a pdf by setting `file`.

Usage

```r
plotTopic(
  object,
  ldaresult, 
  ldaID, 
  select = 1:nrow(ldaresult$document_sums),
  tnames, 
  rel = FALSE, 
  mark = TRUE, 
  unit = "month", 
  curves = c("exact", "smooth", "both"),
  smooth = 0.05,
  main, 
  xlab, 
  ylab, 
  ylim,
  both.lwd, 
  both.lty, 
  col, 
  legend = ifelse(pages, "onlyLast:topright", "topright"), 
  pages = FALSE, 
  natozero = TRUE, 
  file, 
  ...
)
```
Arguments

object  
**textmeta** object with strictly tokenized text component (character vectors) - such as a result of **cleanTexts**

ldaresult  
The result of a function call **LDAnet**

ldaID  
Character vector of IDs of the documents in ldaresult

select  
Integer: Which topics of ldaresult should be plotted (default: all topics)?

tnames  
Character vector of same length as select - labels for the topics (default are the first returned words of **top.topic.words** from the lda package for each topic)

rel  
Logical: Should counts (FALSE) or proportion (TRUE) be plotted (default: FALSE)?

mark  
Logical: Should years be marked by vertical lines (default: TRUE)?

unit  
Character: To which unit should dates be floored (default: "month")? Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see **round_date**

curves  
Character: Should "exact", "smooth" curve or "both" be plotted (default: "exact")?

smooth  
Numeric: Smoothing parameter which is handed over to **lowess** as f (default: 0.05)

main  
Character: Graphical parameter

xlab  
Character: Graphical parameter

ylab  
Character: Graphical parameter

both.lwd  
Graphical parameter for smoothed values if curves = "both"

both.lty  
Graphical parameter for smoothed values if curves = "both"

col  
Graphical parameter, could be a vector. If curves = "both" the function will for every topicgroup plot at first the exact and then the smoothed curve - this is important for your col order.

legend  
Character: Value(s) to specify the legend coordinates (default: "topright", "onlyLast:topright" for pages = TRUE respectively). If "none" no legend is plotted.

pages  
Logical: Should all curves be plotted in a single plot (default: FALSE)? In addition you could set legend = "onlyLast:<argument>" with <argument> as a character legend argument for only plotting a legend on the last plot of set.

natozero  
Logical: Should NAs be coerced to zeros (default: TRUE)? Only has effect if rel = TRUE.

file  
Character: File path if a pdf should be created

...  
Additional graphical parameters

Value

A plot. Invisible: A dataframe with columns date and tnames with the counts/proportion of the selected topics.
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDAprep(text=poliClean$text, vocab=words10)
LDAresult <- LDAgen(documents=poliLDA, K=10, vocab=words10)

# plot all topics
plotTopic(object=poliClean, ldaresult=LDAresult, ldaID=names(poliLDA))

# plot special topics
plotTopic(object=poliClean, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,4))

## End(Not run)

---

**plotTopicWord**

*Plotting Counts of Topics-Words-Combination over Time (Relative to Words)*

**Description**

Creates a plot of the counts/proportion of specified combination of topics and words. It is important to keep in mind that the baseline for proportions are the sums of words, not sums of topics. See also `plotWordpt`. There is an option to plot all curves in one plot or to create one plot for every curve (see `pages`). In addition the plots can be written to a pdf by setting `file`.

**Usage**

```r
plotTopicWord(
  object,
  docs,
  ldaresult, ldaID,
  wordlist = lda::top.topic.words(ldaresult$topics, 1),
  link = c("and", "or"),
  select = 1:nrow(ldaresult$document_sums),
  tnames, wnames,
  rel = FALSE,
  mark = TRUE,
  unit = "month",
  curves = c("exact", "smooth", "both"),
  smooth = 0.05,
  legend = ifelse(pages, "onlyLast:topright", "topright"),
  pages = FALSE,
  natozero = TRUE,
)```
Arguments

object textmeta object with strictly tokenized text component (Character vectors) - such as a result of cleanTexts
docs Object as a result of LDAprep which was handed over to LDAgen
ldaresult The result of a function call LDAgen with docs as argument
ldaID List of character vectors. Every list element is an 'or' link, every character vector is linked by the argument link. If wordlist is only a character vector it will be coerced to a list of the same length as the vector (see as.list), so that the argument link has no effect. Each character vector as a list element represents one curve in the emerging plot.
link Character: Should the (inner) character vectors of each list element be linked by an "and" or an "or" (default: "and")?
select List of integer vectors: Which topics - linked by an "or" every time - should be taken into account for plotting the word counts/proportion (default: all topics as simple integer vector)?
tnames Character vector of same length as select - labels for the topics (default are the first returned words of
wnames Character vector of same length as wordlist - labels for every group of 'and' linked words top.topic.words from the lda package for each topic)
rel Logical: Should counts (FALSE) or proportion (TRUE) be plotted (default: FALSE)?
mark Logical: Should years be marked by vertical lines (default: TRUE)?
unit Character: To which unit should dates be floored (default: "month")? Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see round_date
curves Character: Should "exact", "smooth" curve or "both" be plotted (default: "exact")?
smooth Numeric: Smoothing parameter which is handed over to lowess as f (default: 0.05)
legend Character: Value(s) to specify the legend coordinates (default: "topright", "onlyLast:topright" for pages = TRUE respectively). If "none" no legend is plotted.
plotTopicWord

pages Logical: Should all curves be plotted in a single plot (default: FALSE)? In addition you could set legend = "onlyLast:<argument>" with <argument> as a character legend argument for only plotting a legend on the last plot of set.

natozero Logical: Should NAs be coerced to zeros (default: TRUE)?

file Character: File path if a pdf should be created

main Character: Graphical parameter

xlab Character: Graphical parameter

ylab Character: Graphical parameter

ylim Graphical parameter

both.lwd Graphical parameter for smoothed values if curves = "both"

both.lty Graphical parameter for smoothed values if curves = "both"

col Graphical parameter, could be a vector. If curves = "both" the function will for every wordgroup plot at first the exact and then the smoothed curve - this is important for your col order.

... Additional graphical parameters

Value

A plot. Invisible: A dataframe with columns date and tnames: wnames with the counts/proportion of the selected combination of topics and words.

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
w <- words10$words[words10$wordtable > 10]
poliLDA <- LDAprep(text=poliClean$text, vocab=w)
LDAresult <- LDAgen(documents=poliLDA, K=10, vocab=w)

# plot topwords from each topic
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA))

# plot one word in different topics
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,3,8), wordlist=c("bush"))

# Differences between plotTopicWord and plotWordpt
par(mfrow=c(2,2))
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,3,8), wordlist=c("bush"), rel=FALSE)
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,3,8), wordlist=c("bush"), rel=FALSE)
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,3,8), wordlist=c("bush"), rel=TRUE)
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), select=c(1,3,8), wordlist=c("bush"), rel=TRUE)
```

plotWordpt

Plots Counts of Topics-Words-Combination over Time (Relative to Topics)

Description

Creates a plot of the counts/proportion of specified combination of topics and words. The plot shows how often a word appears in a topic. It is important to keep in mind that the baseline for proportions are the sums of topics, not sums of words. See also plotTopicWord. There is an option to plot all curves in one plot or to create one plot for every curve (see pages). In addition the plots can be written to a pdf by setting file.

Usage

plotWordpt(
  object,
  docs,
  ldaresult,
  ldaID,
  select = 1:nrow(ldaresult$document_sums),
  link = c("and", "or"),
  wordlist = lda::top.topic.words(ldaresult$topics, 1),
  tnames,
  wnames,
  rel = FALSE,
  mark = TRUE,
  unit = "month",
  curves = c("exact", "smooth", "both"),
  smooth = 0.05,
  legend = ifelse(pages, "onlyLast:topright", "topright"),
  pages = FALSE,
  natozero = TRUE,
  file,
  main,
  xlab,
  ylab,
  ylim,
  both.lwd,
  both.lty,
  col,
  ...
)
plotWordpt

Arguments

object  textmeta object with strictly tokenized text component (character vectors) - e.g. a result of cleanTexts
docs Object as a result of LDAprep which was handed over to LDAgen
ldaresult The result of a function call LDAgen with docs as argument
ldaID Character vector of IDs of the documents in ldaresult
select List of integer vectors. Every list element is an 'or' link, every integer string in a vector is linked by the argument link. If select is only a integer vector it will be coerced to a list of the same length as the vector (see as.list), so that the argument link has no effect. Each integer vector as a list element represents one curve in the outcoming plot
link Character: Should the (inner) integer vectors of each list element be linked by an "and" or an "or" (default: "and")?
wordlist List of character vectors: Which words - always linked by an "or" - should be taken into account for plotting the topic counts/proportion (default: the first top.topic.words per topic as simple character vector)?
tnames Character vector of same length as select - labels for the topics (default are the first returned words of
wnames Character vector of same length as wordlist - labels for every group of 'and' linked words top.topic.words from the lda package for each topic)
rel Logical: Should counts (FALSE) or proportion (TRUE) be plotted (default: FALSE)?
mark Logical: Should years be marked by vertical lines (default: TRUE)?
unit Character: To which unit should dates be floored (default: "month")? Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see round_date
curves Character: Should "exact", "smooth" curve or "both" be plotted (default: "exact")?
smooth Numeric: Smoothing parameter which is handed over to lowess as f (default: 0.05)
legend Character: Value(s) to specify the legend coordinates (default: "topright", "onlyLast:topright" for pages = TRUE respectively). If "none" no legend is plotted.
pages Logical: Should all curves be plotted in a single plot (default: FALSE)? In addition you could set legend = "onlyLast:<argument>" with <argument> as a character legend argument for only plotting a legend on the last plot of set.
natozero Logical: Should NAs be coerced to zeros (default: TRUE)?
file Character: File path if a pdf should be created
main Character: Graphical parameter
xlab Character: Graphical parameter
ylab Character: Graphical parameter
ylim Character: Graphical parameter
both.lwd  Graphical parameter for smoothed values if curves = "both"
both.lty  Graphical parameter for smoothed values if curves = "both"
col      Graphical parameter, could be a vector. If curves = "both" the function will plot for every wordgroup the exact at first and then the smoothed curve - this is important for your col order.
...
Additional graphical parameters

Value
A plot. Invisible: A dataframe with columns date and tnames: wnames with the counts/proportion of the selected combination of topics and words.

Examples
```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDAsvest(text=poliClean$text, vocab=words10)
LDAresult <- LDAsvest(documents=poliLDA, K=10, vocab=words10)
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA))
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA), rel=TRUE)

# Differences between plotTopicWord and plotWordpt
par(mfrow=c(2,2))
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA),
              select=c(1,3,8), wordlist=c("bush"), rel=FALSE)
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA),
           select=c(1,3,8), wordlist=c("bush"), rel=FALSE)
plotTopicWord(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA),
              select=c(1,3,8), wordlist=c("bush"), rel=TRUE)
plotWordpt(object=poliClean, docs=poliLDA, ldaresult=LDAresult, ldaID=names(poliLDA),
           select=c(1,3,8), wordlist=c("bush"), rel=TRUE)

## End(Not run)
```

---

**plotWordSub**

*Plotting Counts/Proportion of Words/Docs in LDA-generated Topic-Subcorpora over Time*

**Description**

Creates a plot of the counts/proportion of words/docs in corpora which are generated by a ldaresult. Therefore an article is allocated to a topic - and then to the topics corpus - if there are enough (see limit and alloc) allocations of words in the article to the corresponding topic. Additionally the corpora are reduced by filterWord and a search-argument. The plot shows counts of subcorpora or if rel = TRUE proportion of subcorpora to its corresponding whole corpus.
Usage

plotWordSub(
    object,
    ldaresult,
    ldaID,
    limit = 10,
    alloc = c("multi", "unique", "best"),
    select = 1:nrow(ldaresult$document_sums),
    tnames,
    search,
    ignore.case = TRUE,
    type = c("docs", "words"),
    rel = TRUE,
    mark = TRUE,
    unit = "month",
    curves = c("exact", "smooth", "both"),
    smooth = 0.05,
    main,
    xlab,
    ylab,
    ylim,
    both.lwd,
    both.lty,
    col,
    legend = "topright",
    natozero = TRUE,
    file,
    ...
)

Arguments

object  textmeta object with strictly tokenized text component (character vectors) - such as a result of cleanTexts
ldaresult  The result of a function call LDAGen
ldaID  Character vector of IDs of the documents in ldaresult
limit  Integer/numeric: How often a word must be allocated to a topic to count these article as belonging to this topic - if 0<limit<1 proportion is used (default: 10)?
alloc  Character: Should every article be allocated to multiple topics ("multi"), or maximum one topic ("unique"), or the most representative - exactly one - topic ("best") (default: "multi")? If alloc = "best" limit has no effect.
select  Integer vector: Which topics of ldaresult should be plotted (default: all topics)?
tnames  Character vector of same length as select - labels for the topics (default are the first returned words of top.topic.words from the lda package for each topic)
search  See filterWord
ignore.case  See `filterWord`  

type  Character: Should counts/proportion of documents, where every "docs" or words "words" be plotted (default: "docs")?  

rel  Logical. Should counts (FALSE) or proportion (TRUE) be plotted (default: TRUE)?  

mark  Logical: Should years be marked by vertical lines (default: TRUE)?  

unit  Character: To which unit should dates be floored (default: "month")? Other possible units are "bimonth", "quarter", "season", "halfyear", "year", for more units see `round_date`  

curves  Character: Should "exact", "smooth" curve or "both" be plotted (default: "exact")?  

smooth  Numeric: Smoothing parameter which is handed over to `lowess` as f (default: 0.05)  

main  Character: Graphical parameter  

xlab  Character: Graphical parameter  

ylab  Character: Graphical parameter  

ylim  Graphical parameter (default if rel = TRUE: c(0,1))  

both.lwd  Graphical parameter for smoothed values if curves = "both"  

both.lty  Graphical parameter for smoothed values if curves = "both"  

col  Graphical parameter, could be a vector. If curves = "both" the function will for every wordgroup plot at first the exact and then the smoothed curve - this is important for your col order.  

legend  Character: Value(s) to specify the legend coordinates (default: "topright"). If "none" no legend is plotted.  

natozero  Logical. Should NAs be coerced to zeros (default: TRUE)? Only has effect if rel = TRUE.  

file  Character: File path if a pdf should be created  

...  Additional graphical parameters  

Value  

A plot. Invisible: A dataframe with columns date and tnames with the counts/proportion of the selected topics.  

Examples  

```r  
## Not run:  
data(politics)  
polClean <- cleanTexts(politics)  
poliPresidents <- filterWord(object=poliClean, search=c("bush", "obama"))  
words10 <- makeWordList(text=poliPresidents$text)  
words10 <- words10$words[words10$wordtable > 10]  
poliLDA <- LDAprep(text=poliPresidents$text, vocab=words10)  
LDAreSult <- LDAgen(documents=poliLDA, K=5, vocab=words10)  
plotWordSub(object=poliClean, ldaresult=LDAreSult, ldaID=names(poliLDA), search="obama")  
```  

## End(Not run)  

**Description**

Estimates Precision and Recall for sampling in different intersections

**Usage**

```
precision(w, p, subset)
vprecision(w, p, subset, n)
recall(w, p, subset)
vrecall(w, p, subset, n)
```

**Arguments**

- `w` Numeric vector: Each entry represents one intersection. Proportion of texts in this intersection.
- `p` Numeric vector: Each entry represents one intersection. Proportion of relevant texts in this intersection.
- `subset` Logical vector: Each entry represents one intersection. Controls if the intersection belongs to the subcorpus of interest or not.
- `n` Integer vector: Number of Texts labeled in the corresponding intersection.

**Value**

Estimator for precision, recall, and their variances respectively.

**Examples**

```r
w <- c(0.5, 0.1, 0.2, 0.2)
p <- c(0.01, 0.8, 0.75, 0.95)
subset <- c(FALSE, TRUE, FALSE, TRUE)
n <- c(40, 20, 15, 33)
precision(w, p, subset)
vprecision(w, p, subset, n)
recall(w, p, subset)
vrecall(w, p, subset, n)
```
Description

Reads CSV-files and seperates the text and meta data. The result is a \texttt{textmeta} object.

Usage

\begin{verbatim}
readTextmeta(
  path, file, cols,
  dateFormat = "%Y-%m-%d",
  idCol = "id",
  dateCol = "date",
  titleCol = "title",
  textCol = "text",
  encoding = "UTF-8",
  xmlAction = TRUE,
  duplicateAction = TRUE
)
\end{verbatim}

\begin{verbatim}
readTextmeta.df(
  df, cols = colnames(df),
  dateFormat = "%Y-%m-%d",
  idCol = "id",
  dateCol = "date",
  titleCol = "title",
  textCol = "text",
  xmlAction = TRUE,
  duplicateAction = TRUE
)
\end{verbatim}

Arguments

- \texttt{path} character/data.frame string with path where the data files are OR parameter \texttt{df} for \texttt{readTextmeta.df}
- \texttt{file} character string with names of the CSV files
- \texttt{cols} character vector with columns which should be kept
- \texttt{dateFormat} character string with the date format in the files for \texttt{as.Date}
- \texttt{idCol} character string with column name of the IDs
- \texttt{dateCol} character string with column name of the Dates
- \texttt{titleCol} character string with column name of the Titles
**readWhatsApp**

```r
readWhatsApp(path, file)
```

**Arguments**

- `path` Character: string with path where the data files are. If only `path` is given, `file` will be determined by searching for html files with `list.files` and recursion.
- `file` Character: string with names of the HTML files.

**Value**

- `textmeta` object.

**Description**

Reads HTML-files from WhatsApp and separates the text and meta data.

**Usage**

```r
readWhatsApp(path, file)
```

**Author(s)**

Jonas Rieger (<jonas.rieger@tu-dortmund.de>)
readWiki  Read Pages from Wikipedia

Description

Downloads pages from Wikipedia and extracts some meta information with functions from the package Wikipedia. Creates a textmeta object including the requested pages.

Usage

readWiki(
  category,
  subcategories = TRUE,
  language = "en",
  project = "wikipedia"
)

Arguments

category character articles of which category should be downloaded, see pages_in_category, argument categories
subcategories logical (default: TRUE) should subcategories be downloaded as well
language character (default: "en"), see pages_in_category
project character (default: "wikipedia"), see pages_in_category

Value

textmeta object

Examples

## Not run: corpus <- readWiki(category="Person_(Studentenbewegung)",
subcategories = FALSE, language = "de", project = "wikipedia")
## End(Not run)

readWikinews  Read files from Wikinews

Description

Usage

readWikinews(
    path = getwd(),
    file = list.files(path = path, pattern = "*.xml$", full.names = FALSE, recursive = TRUE)
)

Arguments

path   Path where the data files are.
file   Character string with names of the HTML files.

Value
textmeta-object

---

**removeXML**

*Removes XML/HTML Tags and Umlauts*

Description

Removes XML tags (removeXML), remove or resolve HTML tags (removeHTML) and changes
 german umlauts in a standardized form (removeUmlauts).

Usage

removeXML(x)

removeUmlauts(x)

removeHTML(
    x,
    dec = TRUE,
    hex = TRUE,
    entity = TRUE,
    symbolList = c(1:4, 9, 13, 15, 16),
    delete = TRUE,
    symbols = FALSE
)

Arguments

x   Character: Vector or list of character vectors.
dec Logical: If TRUE HTML-entities in decimal-style would be resolved.
hex Logical: If TRUE HTML-entities in hexadecimal-style would be resolved.
entity Logical: If TRUE HTML-entities in text-style would be resolved.
symbolList numeric vector to choose from the 16 ISO-8859 Lists (ISO-8859 12 did not exist and is empty).
dele Logical: If TRUE all not resolved HTML-entities would be deleted?

Details

The decision which u.type is used should consider the language of the corpus, because in some languages the replacement of umlauts can change the meaning of a word. To change which columns are used by removeXML use argument xmlAction in readTextmeta.

Value

Adjusted character string or list, depending on input.

Examples

xml <- "<text>Some <b>important</b> text</text>"
removeXML(xml)

x <- "&shy; &#248; &oslash;"
removeHTML(x=x, symbolList = 1, dec=TRUE, hex=FALSE, entity=FALSE, delete = FALSE)
removeHTML(x=x, symbolList = c(1,3))

y <- c("Bl\UFCChende Apfelb\UE4ume")
removeUmlauts(y)

description

Sample Texts

Sample texts from different subsets to minimize variance of the recall estimator

Usage

sampling(id, corporaID, label, m, randomize = FALSE, exact = FALSE)

Arguments

id Character: IDs of all texts in the corpus.
corporaID List of Character: Each list element is a character vector and contains the IDs belonging to one subcorpus. Each ID has to be in id.
showMeta

label

Named Logical: Labeling result for already labeled texts. Could be empty, if no labeled data exists. The algorithm sets \( p = 0.5 \) for all intersections. Names have to be id.

m

Integer: Number of new samples.

randomize

Logical: If TRUE calculated split is used as parameter to draw from a multinomial distribution.

exact

Logical: If TRUE exact calculation is used. For the default FALSE an approximation is used.

Value

Character vector of IDs, which should be labeled next.

Examples

```r
id <- paste0("ID", 1:1000)
corporaID <- list(sample(id, 300), sample(id, 100), sample(id, 700))
label <- sample(as.logical(0:1), 150, replace=TRUE)
names(label) <- c(sample(id, 100), sample(corporaID[[2]], 50))
m <- 100
sampling(id, corporaID, label, m)
```

showMeta

Export Readable Meta-Data of Articles.

Description

Exports requested meta-data of articles for given id’s.

Usage

```r
showMeta(
  meta,
  id = meta$id,
  cols = colnames(meta),
  file,
  fileEncoding = "UTF-8"
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta</td>
<td>A data.frame of meta-data as a result of a read-function.</td>
</tr>
<tr>
<td>id</td>
<td>Character vector or matrix including article ids.</td>
</tr>
<tr>
<td>cols</td>
<td>Character vector including the requested columns of meta.</td>
</tr>
<tr>
<td>file</td>
<td>Character Filename for the export.</td>
</tr>
<tr>
<td>fileEncoding</td>
<td>character string: declares file encoding. For more information see write.csv</td>
</tr>
</tbody>
</table>
Value

A list of the requested meta data. If file is set, writes a csv including the meta-data of the requested meta data.

Examples

```r
meta <- data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE)

extractedMeta <- showMeta(meta=meta, cols = c("title", "date"))
```

showTexts

Exports Readable Text Lists

Description

Exports the article id, text, title and date.

Usage

```r
showTexts(object, id = names(object$text), file, fileEncoding = "UTF-8")
```

Arguments

- `object` textmeta object
- `id` Character vector or matrix including article ids
- `file` Character Filename for the export. If not specified the functions output ist only invisible.
- `fileEncoding` character string: declares file encoding. For more information see `write.csv`

Value

A list of the requested articles. If file is set, writes a csv including the meta-data of the requested articles.

Examples

```r
texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
```
Description

Creates, Tests, Summarises and Plots Textmeta-Objects

Usage

textmeta(meta = NULL, text = NULL, metamult = NULL, dateFormat = "%Y-%m-%d")

is.textmeta(x)

## S3 method for class 'textmeta'
print(x, ...)

## S3 method for class 'textmeta'
summary(object, listnames = names(object), metavariables = character(), ...)

## S3 method for class 'textmeta'
plot(x, ...)

Arguments

meta       Data.frame (or matrix) of the meta-data, e.g. as received from as.meta
text      Named list (or character vector) of the text-data (names should correspond to
           IDs in meta)
metamult    List of the metamult-data
dateFormat Character string with the date format in meta for as.Date
x          an R Object.
...         further arguments in plot. Not implemented for print and summary.
object     textmeta object
listnames   Character vector with names of textmeta lists (meta, text, metamult). Summaries
            are generated for those lists only. Default gives summaries for all lists.
metavariables Character vector with variable-names from the meta dataset. Summaries are
            generated for those variables only.

Value

A textmeta object.
**Examples**

```r
texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery
in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

print(corpus)
summary(corpus)
str(corpus)
```

---

**tidy.textmeta**  
*Transform textmeta to an object with tidy text data*

**Description**

Transfers data from a text component of a `textmeta` object to a tidy data frame.

**Usage**

```r
tidy.textmeta(object)

is.textmeta_tidy(x)

## S3 method for class 'textmeta_tidy'
print(x, ...)  
```

**Arguments**

- **object**  
  A `textmeta` object

- **x**  
  an R Object.

- **...**  
  further arguments passed to or from other methods.

**Value**

An object with tidy text data
Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")

obj <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)
tidy.textmeta(obj)

obj <- cleanTexts(obj)
tidy.textmeta(obj)

---

topicCoherence Calculating Topic Coherence

Description

Implementationof Mimno's topic coherence.

Usage

topicCoherence(
  ldaresult,
  documents,
  num.words = 10,
  by.score = TRUE,
  sym.coherence = FALSE,
  epsilon = 1
)

Arguments

ldaresult The result of a function call LDAgen
documents A list prepared by LDAprep.
num.words Integer: Number of topwords used for calculating topic coherence (default: 10).
by.score Logical: Should the Score from top.topic.words be used (default: TRUE)?
sym.coherence Logical: Should a symmetric version of the topic coherence used for the calculations? If TRUE the denominator of the topic coherence uses both wordcounts and not just one.
epsilon Numeric: Smoothing factor to avoid log(0). Default is 1. Stevens et al. recommend a smaller value.
Value

A vector of topic coherences. the length of the vector corresponds to the number of topics in the model.

References


Examples

```r
texts <- list(A="Give a Man a Fish, and You Feed Him for a Day. Teach a Man To Fish, and You Feed Him for a Lifetime", B="So Long, and Thanks for All the Fish", C="A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals.")
corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"), title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"), date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"), additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)
corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
ldaPrep <- LDAprep(text=corpus$text, vocab=wordlist$words)
result <- LDAgen(documents=ldaPrep, K = 3L, vocab=wordlist$words, num.words=3)
topicCoherence(ldaresult=result, documents=ldaPrep, num.words=5, by.score=TRUE)
```

---

**topicsInText**

*Coloring the words of a text corresponding to topic allocation*

**Description**

The function creates a HTML document with the words of texts colored depending on the topic allocation of each word.

**Usage**

```r
topicsInText(
  text,
  ldaID,
  id,
  ldaresult,
)```
label = NULL,
vocab,
wordOrder = c("both", "alphabetical", "topics", ","),
colors = NULL,
fixColors = FALSE,
meta = NULL,
originaltext = NULL,
unclearTopicAssignment = TRUE,
htmlreturn = FALSE
)

Arguments

- **text**: The result of LDAprep
- **ldaID**: List of IDs for text
- **id**: ID of the article of interest
- **ldaresult**: A result object from the standardLDA
- **label**: Optional label for each topic
- **vocab**: Character: Vector of vocab corresponding to the text object
- **wordOrder**: Type of output: "alphabetical" prints the words of the article in alphabetical order, "topics" sorts by topic (biggest topic first) and "both" prints both versions. All other inputs will result to no output (this makes only sense in combination with originaltext.
- **colors**: Character vector of colors. If the vector is shorter than the number of topics it will be completed by "black" entries.
- **fixColors**: Logical: If FALSE the first color will be used for the biggest topic and so on. If fixColors=TRUE the the color-entry corresponding to the position of the topic is choosen.
- **meta**: Optional input for meta data. It will be printed in the header of the output.
- **originaltext**: Optional a list of texts (the text list of the textmeta object) including the desired text. Listnames must be IDs. Necessary for output in original text
- **unclearTopicAssignment**: Logical: If TRUE all words which are assigned to more than one topic will not be colored. Otherwise the words will be colored in order of topic apperance in the ldaresult.
- **htmlreturn**: Logical: HTML output for tests

Value

A HTML document

Examples

```r
## Not run:
data(politics)
poliClean <- cleanTexts(politics)
```
words10 <- makeWordlist(text=poliClean$text)
words10 <- words10$words[words10$wordtable > 10]
poliLDA <- LDAprep(text=poliClean$text, vocab=words10)
LDAresult <- LDAgen(documents=poliLDA, K=10, vocab=words10)
topicsInText(text=politics$text, ldaID=names(poliLDA), id="ID2756",
             ldaresult=LDAresult, vocab=words10)
## End(Not run)

---

**topTexts**  
*Get The IDs Of The Most Representative Texts*

**Description**

The function extracts the text IDs belonging to the texts with the highest relative or absolute number of words per topic.

**Usage**

```r
topTexts(
  ldaresult,  
  ldaID,     
  limit = 20L,  
  rel = TRUE,   
  select = 1:nrow(ldaresult$document_sums),  
  tnames,  
  minlength = 30L
)
```

**Arguments**

- `ldaresult`: LDA result
- `ldaID`: Vector of text IDs
- `limit`: Integer: Number of text IDs per topic.
- `rel`: Logical: Should be the relative frequency be used?
- `select`: Which topics should be returned?
- `tnames`: Names of the selected topics
- `minlength`: Minimal total number of words a text must have to be included

**Value**

Matrix of text IDs.
Examples

texts <- list(A="Give a Man a Fish, and You Feed Him for a Day.
Teach a Man To Fish, and You Feed Him for a Lifetime",
B="So Long, and Thanks for All the Fish",
C="A very able manipulative mathematician, Fisher enjoys a real mastery
in evaluating complicated multiple integrals.")

corpus <- textmeta(meta=data.frame(id=c("A", "B", "C", "D"),
title=c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
date=c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
additionalVariable=1:4, stringsAsFactors=FALSE), text=texts)

corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
ldaPrep <- LDprep(text=corpus$text, vocab=wordlist$words)

LDA <- LDagen(documents=ldaPrep, K = 3L, vocab=wordlist$words, num.words=3)
topTexts(ldaresult=LDA, ldaID=c("A","B","C"), limit = 1L, minlength=2)

topWords

Top Words per Topic

Description

Determines the top words per topic as top.topic.words do. In addition, it is possible to request the
values that are taken for determining the top words per topic. Therefore, the function importance
is used, which also can be called independently.

Usage

topWords(topics, numWords = 1, byScore = TRUE, epsilon = 1e-05, values = FALSE)

importance(topics, epsilon = 1e-05)

Arguments

topics named matrix: The counts of vocabularies (column wise) in topics (row wise).
numWords integer(1): The number of requested top words per topic.
byScore logical(1): Should the values that are taken for determining the top words per
topic be calculated by the function importance (TRUE) or should the absolute
counts be considered (FALSE)?
epsilon numeric(1): Small number to add to logarithmic calculations to overcome the
issue of determining log(0).
values logical(1): Should the values that are taken for determining the top words per
topic be returned?
Value

Matrix of top words or, if value is TRUE a list of matrices with entries word and val.

Examples

texts <- list(
    A = "Give a Man a Fish, and You Feed Him for a Day.
    Teach a Man To Fish, and You Feed Him for a Lifetime",
    B = "So Long, and Thanks for All the Fish",
    C = "A very able manipulative mathematician, Fisher enjoys a real mastery in evaluating complicated multiple integrals."
)
corpus <- textmeta(meta = data.frame(id = c("A", "B", "C", "D"),
    title = c("Fishing", "Don't panic!", "Sir Ronald", "Berlin"),
    date = c("1885-01-02", "1979-03-04", "1951-05-06", "1967-06-02"),
    additionalVariable = 1:4, stringsAsFactors = FALSE), text = texts)
corpus <- cleanTexts(corpus)
wordlist <- makeWordlist(corpus$text)
ldaPrep <- LDAprep(text = corpus$text, vocab = wordlist$words)
LDA <- LDAgen(documents = ldaPrep, K = 3L, vocab = wordlist$words, num.words = 3)
topWords(LDA$topics)
importance(LDA$topics)
## Index

* **manip**
  - as.corpus.textmeta, 3
  - as.meta, 4
  - as.textmeta.corpus, 5
  - cleanTexts, 6
  - duplist, 10
  - filterDate, 13
  - LDAprep, 22
  - makeWordlist, 23
  - mergeLDA, 24
  - mergeTextmeta, 25
  - readTextmeta, 44
  - readWhatsApp, 45
  - readWiki, 46
  - removeXML, 47
  - showMeta, 49
  - showTexts, 50
  - textmeta, 51
  - tidy.textmeta, 52
  - topTexts, 56

  as.corpus.textmeta, 3
  as.Date, 4, 5, 44, 51
  as.list, 28, 36, 39
  as.meta, 4, 51
  as.textmeta.corpus, 5

  cleanTexts, 6, 22, 28, 34, 36, 39, 41
  clusterTopics, 7, 24
  corpus, 3, 5

  deleteAndRenameDuplicates, 5, 9, 45
  docnames, 3
  docvars, 3
  duplist, 10

  filterCount, 12
  filterDate, 13
  filterID, 14
  filterWord, 15, 40–42

  grepl, 28
  hclust, 8
  heatmap, 31

  importance (topWords), 57
  intersection, 25
  intruderTopics, 17
  intruderWords, 19
  is.duplist (duplist), 10
  is.textmeta (textmeta), 51
  is.textmeta_tidy ( tidy.textmeta), 52

  lda, 22
  lda.collapsed.gibbs.sampler, 21
  LDAGen, 8, 18, 19, 21, 33, 34, 36, 39, 41, 53
  LDAprep, 21, 22, 36, 39, 53, 55
  list.files, 45
  lowess, 29, 32, 34, 36, 39, 42

  makeWordlist, 23
  mergeLDA, 24
  mergeTextmeta, 25

  pages_in_category, 46
  pdf, 8
  plot, 8
  plot.textmeta (textmeta), 51
  plotArea, 26
  plotFreq, 27
  plotHeat, 30
  plotScot, 31
  plotTopic, 33
  plotTopicWord, 35, 38
  plotWordpt, 35, 38
  plotWordSub, 40
  precision, 43
  print.duplist (duplist), 10
  print.textmeta (textmeta), 51
  print.textmeta_tidy ( tidy.textmeta), 52
quanteda, 3, 5

readTextmeta, 44, 48
readWhatsApp, 45
readWiki, 46
readWikinews, 46
recall (precision), 43
removeHTML (removeXML), 47
removePunctuation, 7
removeUmlauts (removeXML), 47
removeXML, 45, 47
round_date, 26, 29, 30, 32, 34, 36, 39, 42

sampling, 48
showMeta, 49
showTexts, 50
stopwords, 6
summary.duplist (duplist), 10
summary.textmeta (textmeta), 51

textmeta, 3, 5–7, 12, 14–16, 18, 25, 28, 30, 32, 34, 36, 39, 41, 44–46, 50, 51, 52
tidy.textmeta, 52
top.topic.words, 34, 36, 39, 41, 53, 57
topicCoherence, 53
topicsInText, 54
topTexts, 56
topWords, 57

union, 25

vprecision (precision), 43
vrecall (precision), 43

WikipediaR, 46
write.csv, 49, 50