Package 'tidytext'

June 19, 2017

Type Package

Title Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools

Version 0.1.3

Description Text mining for word processing and sentiment analysis using 'dplyr', 'ggplot2', and other tidy tools.

License MIT + file LICENSE

LazyData TRUE

URL http://github.com/juliasilge/tidytext

BugReports http://github.com/juliasilge/tidytext/issues

RoxygenNote 6.0.1

Depends R (>= 2.10)

Imports dplyr, stringr, hunspell, broom, Matrix, tokenizers, janeaustenr, purrr (>= 0.1.1), methods

Suggests readr, tidyr, XML, tm, quanteda, wordcloud, knitr, rmarkdown, ggplot2, reshape2, topicmodels, NLP, scales, gutenbergr, testthat, mallet

VignetteBuilder knitr

NeedsCompilation no

Author Gabriela De Queiroz [ctb], Oliver Keyes [ctb], David Robinson [aut], Julia Silge [aut, cre]

Maintainer Julia Silge <julia.silge@gmail.com>

Repository CRAN

Date/Publication 2017-06-19 14:42:14 UTC

R topics documented:

`bind_tf_idf` *Bind the term frequency and inverse document frequency of a tidy text dataset to the dataset*

Description

Calculate and bind the term frequency and inverse document frequency of a tidy text dataset, along with the product, tf-idf to the dataset. Each of these values are added as columns.

Usage

```
bind_tf_idf(tbl, term_col, document_col, n_col)

bind_tf_idf_(tbl, term_col, document_col, n_col)
```

Arguments

`tbl`	A tidy text dataset with one-row-per-term-per-document
`term_col`	Column containing terms
`document_col`	Column containing document IDs
`n_col`	Column containing document-term counts

Details

`tf_idf` is given bare names, while `tf_idf_` is given strings and is therefore suitable for programming with.

If the dataset is grouped, the groups are ignored but are retained.

The dataset must have exactly one row per document-term combination for this to work.

Examples

```
library(dplyr)
library(janeaustenr)

book_words <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word, sort = TRUE) %>%
  ungroup()

book_words

# find the words most distinctive to each document
book_words %>%
  bind_tf_idf(word, book, n) %>%
  arrange(desc(tf_idf))
```

`cast_sparse` *Create a sparse matrix from row names, column names, and values in a table.*

Description

Create a sparse matrix from row names, column names, and values in a table.

Usage

```
cast_sparse(data, row, column, value)
```

Arguments

`data`	A tbl
`row`	A bare column name to use as row names in sparse matrix
`column`	A bare column name to use as column names in sparse matrix
`value`	A bare column name to use as sparse matrix values, default 1

Details

Note that cast_sparse ignores groups in a grouped tbl_df.

Value

A sparse Matrix object, with one row for each unique value in the `row` column, one column for each unique value in the `column` column, and with as many non-zero values as there are rows in `data`.

Examples

```
dat <- data.frame(a = c("row1", "row1", "row2", "row2", "row2"),
                  b = c("col1", "col2", "col1", "col3", "col4"),
                  val = 1:5)

cast_sparse(dat, a, b)

cast_sparse(dat, a, b, val)
```

---

cast_sparse_ *Standard-evaluation version of cast_sparse*

Description

Standard-evaluation version of cast_sparse

Usage

```
cast_sparse_(data, row_col, column_col, value_col = 1, ...)
```

Arguments

`data`	A tbl
`row_col`	String version of column to use as row names
`column_col`	String version of column to use as column names
`value_col`	String version of column to use as sparse matrix values, or a numeric vector to use. Default 1 (to create a binary matrix)
`...`	Extra arguments to pass on to `sparseMatrix`

cast_tdm_ *Casting a data frame to a DocumentTermMatrix, TermDocumentMatrix, or dfm*

Description

This turns a "tidy" one-term-per-document-per-row data frame into a DocumentTermMatrix or TermDocumentMatrix from the tm package, or a dfm from the quanteda package. Each caster can be called either with non-standard evaluation (bare column names) or character vectors (for `cast_tdm_` and `cast_dtm_`). It ignores groups.

Usage

```
cast_tdm_(data, term_col, document_col, value_col, weighting = tm::weightTf,
  ...)

cast_tdm(data, term, document, value, weighting = tm::weightTf, ...)

cast_dtm_(data, document_col, term_col, value_col, weighting = tm::weightTf,
  ...)

cast_dtm(data, document, term, value, weighting = tm::weightTf, ...)

cast_dfm_(data, document_col, term_col, value_col, ...)

cast_dfm(data, document, term, value, ...)
```

Arguments

`data`	Table with one-term-per-document-per-row
`weighting`	The weighting function for the DTM/TDM (default is term-frequency, effectively unweighted)
`...`	Extra arguments passed on to `sparseMatrix`
`term, term_col`	(Bare) name of a column with terms
`document, document_col`	(Bare) name of a column with documents
`value, value_col`	(Bare) name of a column containing values

corpus_tidiers *Tidiers for a corpus object from the quanteda package*

Description

Tidy a corpus object from the quanteda package. `tidy` returns a tbl_df with one-row-per-document, with a `text` column containing the document's text, and one column for each document-level metadata. `glance` returns a one-row tbl_df with corpus-level metadata, such as source and created. For Corpus objects from the tm package, see `tidy.Corpus`.

Usage

```
## S3 method for class 'corpus'
tidy(x, ...)

## S3 method for class 'corpus'
glance(x, ...)
```

Arguments

x	A Corpus object, such as a VCorpus or PCorpus
...	Extra arguments, not used

Details

For the most part, the `tidy` output is equivalent to the "documents" data frame in the corpus object, except that it is converted to a tbl_df, and `texts` column is renamed to `text` to be consistent with other uses in tidytext.

Similarly, the `glance` output is simply the "metadata" object, with NULL fields removed and turned into a one-row tbl_df.

Examples

```
if (requireNamespace("quanteda", quietly = TRUE)) {
 data("data_corpus_inaugural", package = "quanteda")

 data_corpus_inaugural

 tidy(data_corpus_inaugural)
}
```

dictionary_tidiers *Tidy dictionary objects from the quanteda package*

Description

Tidy dictionary objects from the quanteda package

Usage

```
## S3 method for class 'dictionary'
tidy(x, regex = FALSE, ...)
```

Arguments

x	A dictionary object
regex	Whether to turn dictionary items from a glob to a regex
...	Extra arguments, not used

Value

A data frame with two columns: category and word.

get_sentiments *Get a tidy data frame of a single sentiment lexicon*

Description

Get specific sentiment lexicons in a tidy format, with one row per word, in a form that can be joined with a one-word-per-row dataset. Each of these comes from the included sentiments data frame, but this performs the filtering for a specific lexicon, and removes columns that are not used in that lexicon.

Usage

```
get_sentiments(lexicon = c("afinn", "bing", "nrc", "loughran"))
```

Arguments

lexicon	The sentiment lexicon to retrieve; either "afinn", "bing", "nrc", or "loughran"

Value

A tbl_df with a word column, and either a sentiment column (if lexicon is not "afinn") or a numeric score column (if lexicon is "afinn").

Examples

```
library(dplyr)
get_sentiments("afinn")
get_sentiments("bing")
```

`lda_tidiers` *Tidiers for LDA objects from the topicmodels package*

Description

Tidy the results of a Latent Dirichlet Allocation.

Usage

```
## S3 method for class 'LDA'
tidy(x, matrix = c("beta", "gamma"), log = FALSE, ...)

## S3 method for class 'LDA'
augment(x, data, ...)

## S3 method for class 'LDA'
glance(x, ...)
```

Arguments

`x`	An LDA (or LDA_VEM) object from the topicmodels package
`matrix`	Whether to tidy the beta (per-term-per-topic, default) or gamma (per-document-per-topic) matrix
`log`	Whether beta/gamma should be on a log scale, default FALSE
`...`	Extra arguments, not used
`data`	For `augment`, the data given to the LDA function, either as a DocumentTermMatrix or as a tidied table with "document" and "term" columns

Value

`tidy` returns a tidied version of either the beta or gamma matrix.

If `matrix == "beta"` (default), returns a table with one row per topic and term, with columns

topic Topic, as an integer

term Term

beta Probability of a term generated from a topic according to the multinomial model

If `matrix == "gamma"`, returns a table with one row per topic and document, with columns

topic Topic, as an integer

document Document name or ID

gamma Probability of topic given document

`augment` returns a table with one row per original document-term pair, such as is returned by tdm_tidiers:

document Name of document (if present), or index

term Term

.topic Topic assignment

If the `data` argument is provided, any columns in the original data are included, combined based on the `document` and `term` columns.

`glance` always returns a one-row table, with columns

iter Number of iterations used

terms Number of terms in the model

alpha If an LDA_VEM, the parameter of the Dirichlet distribution for topics over documents

Examples

```
if (requireNamespace("topicmodels", quietly = TRUE)) {
  set.seed(2016)
  library(dplyr)
  library(topicmodels)

  data("AssociatedPress", package = "topicmodels")
  ap <- AssociatedPress[1:100, ]
  lda <- LDA(ap, control = list(alpha = 0.1), k = 4)

  # get term distribution within each topic
  td_lda <- tidy(lda)
  td_lda

  library(ggplot2)

  # visualize the top terms within each topic
  td_lda_filtered <- td_lda %>%
    filter(beta > .004) %>%
    mutate(term = reorder(term, beta))

  ggplot(td_lda_filtered, aes(term, beta)) +
    geom_bar(stat = "identity") +
    facet_wrap(~ topic, scales = "free") +
    theme(axis.text.x = element_text(angle = 90, size = 15))

  # get classification of each document
  td_lda_docs <- tidy(lda, matrix = "gamma")
  td_lda_docs
```

```
  doc_classes <- td_lda_docs %>%
    group_by(document) %>%
    top_n(1) %>%
    ungroup()

  doc_classes

  # which were we most uncertain about?
  doc_classes %>%
    arrange(gamma)
}
```

`mallet_tidiers` *Tidiers for Latent Dirichlet Allocation models from the mallet package*

Description

Tidy LDA models fit by the mallet package, which wraps the Mallet topic modeling package in Java. The arguments and return values are similar to `lda_tidiers`.

Usage

```
## S3 method for class 'jobjRef'
tidy(x, matrix = c("beta", "gamma"), log = FALSE,
  normalized = TRUE, smoothed = TRUE, ...)

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

Arguments

`x`	A jobjRef object, of type RTopicModel, such as created by `MalletLDA`.
`matrix`	Whether to tidy the beta (per-term-per-topic, default) or gamma (per-document-per-topic) matrix.
`log`	Whether beta/gamma should be on a log scale, default FALSE
`normalized`	If true (default), normalize so that each document or word sums to one across the topics. If false, values will be integers representing the actual number of word-topic or document-topic assignments.
`smoothed`	If true (default), add the smoothing parameter to each to avoid any values being zero. This smoothing parameter is initialized as `alpha.sum` in `MalletLDA`.
`...`	Extra arguments, not used
`data`	For `augment`, the data given to the LDA function, either as a DocumentTermMatrix or as a tidied table with "document" and "term" columns.

Details

Note that the LDA models from MalletLDA are technically a special case of S4 objects with class jobjRef. These are thus implemented as jobjRef tidiers, with a check for whether the toString output is as expected.

Value

augment must be provided a data argument containing one row per original document-term pair, such as is returned by tdm_tidiers, containing columns document and term. It returns that same data with an additional column .topic with the topic assignment for that document-term combination.

See Also

lda_tidiers, mallet.doc.topics, mallet.topic.words

Examples

```
## Not run:
library(mallet)
library(dplyr)

data("AssociatedPress", package = "topicmodels")
td <- tidy(AssociatedPress)

# mallet needs a file with stop words
tmp <- tempfile()
writeLines(stop_words$word, tmp)

# two vectors: one with document IDs, one with text
docs <- td %>%
  group_by(document = as.character(document)) %>%
  summarize(text = paste(rep(term, count), collapse = " "))

docs <- mallet.import(docs$document, docs$text, tmp)

# create and run a topic model
topic_model <- MalletLDA(num.topics = 4)
topic_model$loadDocuments(docs)
topic_model$train(20)

# tidy the word-topic combinations
td_beta <- tidy(topic_model)
td_beta

# Examine the four topics
td_beta %>%
  group_by(topic) %>%
  top_n(8, beta) %>%
  ungroup() %>%
  mutate(term = reorder(term, beta)) %>%
```

```
  ggplot(aes(term, beta)) +
  geom_col() +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()

# find the assignments of each word in each document
assignments <- augment(topic_model, td)
assignments

## End(Not run)
```

---

parts_of_speech *Parts of speech for English words from the Moby Project*

Description

Parts of speech for English words from the Moby Project by Grady Ward. Words with non-ASCII characters and items with a space have been removed.

Usage

```
parts_of_speech
```

Format

A data frame with 205,985 rows and 2 variables:

word An English word

pos The part of speech of the word. One of 13 options, such as "Noun", "Adverb", "Adjective"

Source

http://icon.shef.ac.uk/Moby/mpos.html

Examples

```
library(dplyr)

parts_of_speech

parts_of_speech %>%
  count(pos, sort = TRUE)
```

`sentiments` *Sentiment lexicons from three sources*

Description

Three lexicons for sentiment analysis are combined here in a tidy data frame. The lexicons are the NRC Emotion Lexicon from Saif Mohammad and Peter Turney, the sentiment lexicon from Bing Liu and collaborators, of Finn Arup Nielsen, and of Tim Loughran and Bill Loughran. Words with non-ASCII characters were removed from the lexicons.

Usage

```
sentiments
```

Format

A data frame with 27,314 rows and 4 variables:

word An English word

sentiment A sentiment whose possible values depend on the lexicon. The "afinn" lexicon has no sentiment category (all are NA), and each of the others can be "positive" or "negative". The NRC lexicon can also be "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", or "trust", and the Loughran lexicon can also be "litigious", "uncertainty", "constraining", and "superfluous".

lexicon The source of the sentiment for the word. One of either "nrc", "bing", or "AFINN".

score A numerical score for the sentiment. This value is `NA` for the Bing and NRC lexicons, and runs between -5 and 5 for the AFINN lexicon.

Details

Note that the loughran lexicon is best suited for financial text, (e.g. where words like "share" is not necessarily positive, and "liability" not necessarily negative).

Source

- http://saifmohammad.com/WebPages/lexicons.html
- https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
- http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
- http://www3.nd.edu/~mcdonald/Word_Lists.html

stop_words *Various lexicons for English stop words*

Description

English stop words from three lexicons, as a data frame. The onix and SMART sets are pulled from the tm package. Note that words with non-ASCII characters have been removed.

Usage

```
stop_words
```

Format

A data frame with 1149 rows and 2 variables:

word An English word

lexicon The source of the stop word. Either "onix", "SMART", or "snowball"

Source

- http://www.lextek.com/manuals/onix/stopwords1.html
- http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop
- http://snowball.tartarus.org/algorithms/english/stop.txt

tdm_tidiers *Tidy DocumentTermMatrix, TermDocumentMatrix, and related objects from the tm package*

Description

Tidy a DocumentTermMatrix or TermDocumentMatrix into a three-column data frame: `term{}`, and value (with zeros missing), with one-row-per-term-per-document.

Usage

```
## S3 method for class 'DocumentTermMatrix'
tidy(x, ...)

## S3 method for class 'TermDocumentMatrix'
tidy(x, ...)

## S3 method for class 'dfmSparse'
tidy(x, ...)
```

```
## S3 method for class 'simple_triplet_matrix'
tidy(x, row_names = NULL, col_names = NULL,
  ...)
```

Arguments

x	A DocumentTermMatrix or TermDocumentMatrix object
...	Extra arguments, not used
row_names	Specify row names
col_names	Specify column names

Examples

```
if (requireNamespace("topicmodels", quietly = TRUE)) {
  data("AssociatedPress", package = "topicmodels")
  AssociatedPress

  tidy(AssociatedPress)
}
```

---

tidy.Corpus *Tidy a Corpus object from the tm package*

Description

Tidy a Corpus object from the tm package. Returns a data frame with one-row-per-document, with a `text` column containing the document's text, and one column for each local (per-document) metadata tag. For corpus objects from the quanteda package, see `tidy.corpus`.

Usage

```
## S3 method for class 'Corpus'
tidy(x, collapse = "\n", ...)
```

Arguments

x	A Corpus object, such as a VCorpus or PCorpus
collapse	A string that should be used to collapse text within each corpus (if a document has multiple lines). Give NULL to not collapse strings, in which case a corpus will end up as a list column if there are multi-line documents.
...	Extra arguments, not used

Examples

```
library(dplyr)   # displaying tbl_dfs

if (requireNamespace("tm", quietly = TRUE)) {
  library(tm)
  #' # tm package examples
  txt <- system.file("texts", "txt", package = "tm")
  ovid <- VCorpus(DirSource(txt, encoding = "UTF-8"),
                  readerControl = list(language = "lat"))

  ovid
  tidy(ovid)

  # choose different options for collapsing text within each
  # document
  tidy(ovid, collapse = "")$text
  tidy(ovid, collapse = NULL)$text

  # another example from Reuters articles
  reut21578 <- system.file("texts", "crude", package = "tm")
  reuters <- VCorpus(DirSource(reut21578),
                     readerControl = list(reader = readReut21578XMLasPlain))
  reuters

  tidy(reuters)
}
```

`tidytext` *tidytext: Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools*

Description

This package implements tidy data principles to make many text mining tasks easier, more effective, and consistent with tools already in wide use.

Details

Much of the infrastructure needed for text mining with tidy data frames already exists in packages like dplyr, broom, tidyr and ggplot2.

In this package, we provide functions and supporting data sets to allow conversion of text to and from tidy formats, and to switch seamlessly between tidy tools and existing text mining packages.

To learn more about tidytext, start with the vignettes: `browseVignettes(package = "tidytext")`

tidy_triplet *Utility function to tidy a simple triplet matrix*

Description

Utility function to tidy a simple triplet matrix

Usage

```
tidy_triplet(x, triplets, row_names = NULL, col_names = NULL)
```

Arguments

x	Object with rownames and colnames
triplets	A data frame or list of i, j, x
row_names	rownames, if not gotten from rownames(x)
col_names	colnames, if not gotten from colnames(x)

unnest_tokens *Split a column into tokens using the tokenizers package*

Description

Split a column into tokens using the tokenizers package, splitting the table into one-token-per-row. unnest_tokens_ is the standard evaluation version.

Usage

```
unnest_tokens(tbl, output, input, token = "words", format = c("text", "man",
  "latex", "html", "xml"), to_lower = TRUE, drop = TRUE, collapse = NULL,
  ...)

unnest_tokens_(tbl, output, input, token = "words", format = c("text",
  "man", "latex", "html", "xml"), to_lower = TRUE, drop = TRUE,
  collapse = NULL, ...)
```

Arguments

tbl	Data frame
output	Output column to be created as bare name.
input	Input column that gets split as bare name.
token	Unit for tokenizing, or a custom tokenizing function. Built-in options are "words" (default), "characters", "ngrams", "skip_ngrams", "sentences", "lines", "paragraphs", and "regex". If a function, should take a character vector and return a list of character vectors of the same length.

format	Either "text", "man", "latex", "html", or "xml". If not text, this uses the hunspell tokenizer, and can tokenize only by "word"
to_lower	Whether to turn column lowercase.
drop	Whether original input column should get dropped. Ignored if the original input and new output column have the same name.
collapse	Whether to combine text with newlines first in case tokens (such as sentences or paragraphs) span multiple lines. If NULL, collapses when token method is "ngrams", "skip_ngrams", "sentences", "lines", "paragraphs", or "regex".
...	Extra arguments passed on to the tokenizer, such as n and k for "ngrams" and "skip_ngrams" or pattern for "regex".

Details

If the unit for tokenizing is ngrams, skip_ngrams, sentences, lines, paragraphs, or regex, the entire input will be collapsed together before tokenizing.

If format is anything other than "text", this uses the hunspell_parse tokenizer instead of the tokenizers package. This does not yet have support for tokenizing by any unit other than words.

Examples

```
library(dplyr)
library(janeaustenr)

d <- data_frame(txt = prideprejudice)
d

d %>%
  unnest_tokens(word, txt)

d %>%
  unnest_tokens(sentence, txt, token = "sentences")

d %>%
  unnest_tokens(ngram, txt, token = "ngrams", n = 2)

d %>%
  unnest_tokens(ngram, txt, token = "skip_ngrams", n = 4, k = 2)

d %>%
  unnest_tokens(chapter, txt, token = "regex", pattern = "Chapter [\\d]")

# custom function
d %>%
  unnest_tokens(word, txt, token = stringr::str_split, pattern = " ")

# tokenize HTML
h <- data_frame(row = 1:2,
                text = c("<h1>Text <b>is<b>", "<a href='example.com'>here</a>"))
```

```
h %>%
  unnest_tokens(word, text, format = "html")
```

Index