

Package ‘tidytext’

October 28, 2016

Type Package

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

Version 0.1.2

Date 2016-10-27

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 5.0.1

Depends R (>= 2.10)

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

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

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2016-10-28 01:17:37

R topics documented:

bind_tf_idf	2
cast_sparse	3
cast_sparse_	4
cast_tdm_	5
corpus_tidiers	6
dictionary_tidiers	7
get_sentiments	7
lda_tidiers	8
pair_count	10
parts_of_speech	11
sentiments	12
stop_words	13
tdm_tidiers	13
tidy.Corpora	14
tidytext	15
tidy_triplet	16
unnest_tokens	16

Index	19
--------------	-----------

bind_tf_idf	<i>Bind the term frequency and inverse document frequency of a tidy text dataset to the dataset</i>
-------------	---

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	<i>Create a sparse matrix from row names, column names, and values in a table.</i>
-------------	--

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_	<i>Casting a data frame to a DocumentTermMatrix, TermDocumentMatrix, or dfm</i>
-----------	---

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

<code>x</code>	A Corpus object, such as a <code>VCorpus</code> or <code>PCorpus</code>
<code>...</code>	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 = FALSE)) {
  data("inaugCorpus", package = "quanteda")

  inaugCorpus

  tidy(inaugCorpus)
}
```

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"))
```

Arguments

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

Value

A `tbl_df` with a word column, and either a sentiment column (if lexicon is "bing" or "nrc") 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

<code>x</code>	An LDA (or LDA_VEM) object from the topicmodels package
<code>matrix</code>	Whether to tidy the beta (per-term-per-topic, default) or gamma (per-document-per-topic) matrix
<code>log</code>	Whether beta/gamma should be on a log scale, default FALSE
<code>...</code>	Extra arguments, not used
<code>data</code>	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

```
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)

```

pair_count	<i>Count pairs of items that cooccur within a group</i>
------------	---

Description

This function has been deprecated. Work on this and similar functions is now going on in the `widyr` package (<https://github.com/dgrtwo/widyr>). Count the number of times pairs of items cooccur within a group. This returns a table with one row for each word-word pair that occurs within a group, along with `n`, the number of groups the pair cooccurs in. `pair_count_` is the standard-evaluation version that can be programmed with.

Usage

```

pair_count(data, group, value, unique_pair = TRUE, self = FALSE,
           sort = FALSE)

pair_count_(data, group_col, value_col, unique_pair = TRUE, self = FALSE,
            sort = FALSE)

```

Arguments

<code>data</code>	A <code>tbl</code>
<code>group, group_col</code>	Column to count pairs within
<code>value, value_col</code>	Column containing values to count pairs of
<code>unique_pair</code>	Whether to have only one pair of <code>value1</code> and <code>value2</code> . Setting this to <code>FALSE</code> is useful if you want to afterwards find the most common values paired with one of interest.
<code>self</code>	Whether to include an item as co-occurring with itself
<code>sort</code>	Whether to sort in decreasing order of frequency

Value

A data frame with three columns: `value1`, `value2`, and `n`.

Examples

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

pride_prejudice_words <- data_frame(text = prideprejudice) %>%
  mutate(line = row_number()) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

# find words that co-occur within lines
pride_prejudice_words %>%
  pair_count(line, word, sort = TRUE)

# when finding words most often occurring with a particular word,
# use unique_pair = FALSE
pride_prejudice_words %>%
  pair_count(line, word, sort = TRUE, unique_pair = FALSE) %>%
  filter(value1 == "elizabeth")

## 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, and the lexicon of Finn Arup Nielsen. Words with non-ASCII characters were removed from the lexicons.

Usage

```
sentiments
```

Format

A data frame with 23,165 rows and 4 variables:

word An English word

sentiment One of either positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, trust, or NA. The Bing lexicon has positive/negative, the NRC lexicon has all options except NA, and the AFINN lexicon has only NA.

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.

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

stop_words	<i>Various lexicons for English stop words</i>
------------	--

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	<i>Tidy DocumentTermMatrix, TermDocumentMatrix, and related objects from the tm package</i>
-------------	---

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	<i>Utility function to tidy a simple triplet matrix</i>
--------------	---

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	<i>Split a column into tokens using the tokenizers package</i>
---------------	--

Description

Split a column into tokens using the tokenizers package

Usage

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

```
unnest_tokens(tbl, output, input, token = "words", to_lower = TRUE,
  drop = TRUE, collapse = NULL, ...)
```

Arguments

tbl	Data frame
output_col	Output column to be created
input_col	Input column that gets split
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"
output	Output column to be created as bare name
input	Input column that gets split as bare name

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(janeaugenr)

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

*Topic **datasets**

- parts_of_speech, [11](#)
 - sentiments, [12](#)
 - stop_words, [13](#)
- augment.LDA (lda_tidiers), [8](#)
- bind_tf_idf, [2](#)
- bind_tf_idf_ (bind_tf_idf), [2](#)
- cast_dfm (cast_tdm_), [5](#)
- cast_dfm_ (cast_tdm_), [5](#)
- cast_dtm (cast_tdm_), [5](#)
- cast_dtm_ (cast_tdm_), [5](#)
- cast_sparse, [3](#)
- cast_sparse_, [4](#)
- cast_tdm (cast_tdm_), [5](#)
- cast_tdm_, [5](#)
- corpus_tidiers, [6](#)
- dictionary_tidiers, [7](#)
- get_sentiments, [7](#)
- glance.corpus (corpus_tidiers), [6](#)
- glance.LDA (lda_tidiers), [8](#)
- hunspell_parse, [17](#)
- lda_tidiers, [8](#)
- pair_count, [10](#)
- pair_count_ (pair_count), [10](#)
- parts_of_speech, [11](#)
- sentiments, [7](#), [12](#)
- sparseMatrix, [4](#), [5](#)
- stop_words, [13](#)
- tdm_tidiers, [9](#), [13](#)
- tidy.Corpus, [6](#), [14](#)
- tidy.corpus, [14](#)
- tidy.corpus (corpus_tidiers), [6](#)
- tidy.dfmSparse (tdm_tidiers), [13](#)
- tidy.dictionary (dictionary_tidiers), [7](#)
- tidy.DocumentTermMatrix (tdm_tidiers), [13](#)
- tidy.LDA (lda_tidiers), [8](#)
- tidy.simple_triplet_matrix (tdm_tidiers), [13](#)
- tidy.TermDocumentMatrix (tdm_tidiers), [13](#)
- tidy_triplet, [16](#)
- tidytext, [15](#)
- tidytext-package (tidytext), [15](#)
- unnest_tokens, [16](#)
- unnest_tokens_ (unnest_tokens), [16](#)