Package ‘tidytext’

July 29, 2019

Type Package
Title Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools
Version 0.2.2
Description Text mining for word processing and sentiment analysis using 'dplyr', 'ggplot2', and other tidy tools.
License MIT + file LICENSE
Encoding UTF-8
LazyData TRUE
URL http://github.com/juliasilge/tidytext
BugReports http://github.com/juliasilge/tidytext/issues
RoxygenNote 6.1.1
Depends R (>= 2.10)
Imports rlang, dplyr, stringr, hunspell, generics, Matrix, tokenizers, janeaustenr, purrr (>= 0.1.1), methods, stopwords
Suggests readr, tidyr, XML, tm, quanteda, knitr, rmarkdown, ggplot2, reshape2, wordcloud, topicmodels, NLP, scales, gutenbergr, testthat, vdiffrr, malley, stm, data.table, broom, textdata
VignetteBuilder knitr
NeedsCompilation no
Author Gabriela De Queiroz [ctb], Emil Hvitfeldt [ctb], Os Keyes [ctb] (https://orcid.org/0000-0001-5196-609X), Kanishka Misra [ctb], Tim Mastny [ctb], Jeff Erickson [ctb], David Robinson [aut], Julia Silge [aut, cre] (https://orcid.org/0000-0002-3671-836X)
Maintainer Julia Silge <julia.silge@gmail.com>
Repository CRAN
Date/Publication 2019-07-29 13:30:02 UTC
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. This function supports non-standard evaluation through the tidyeval framework.

Usage

```r
bind_tf_idf(tbl, term, document, n)
```

Arguments

- `tbl`: A tidy text dataset with one-row-per-term-per-document
- `term`: Column containing terms as string or symbol
- `document`: Column containing document IDs as string or symbol
- `n`: Column containing document-term counts as string or symbol
### cast_sparse

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

#### Description

This function supports non-standard evaluation through the tidyeval framework.

#### Usage

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

#### Arguments

- `data`: A tbl
- `row`: Column name to use as row names in sparse matrix, as string or symbol
- `column`: Column name to use as column names in sparse matrix, as string or symbol
- `value`: Column name to use as sparse matrix values (default 1) as string or symbol
- `...`: Extra arguments to pass on to `sparseMatrix`
Details

Note that cast_sparse ignores groups in a grouped tbl_df. The arguments row, column, and value are passed by expression and support quasiquotation; you can unquote strings and symbols.

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_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. These functions support non-standard evaluation through the tidyeval framework. Groups are ignored.

Usage

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

Arguments

data Table with one-term-per-document-per-row
term Column containing terms as string or symbol
document Column containing document IDs as string or symbol
value Column containing values as string or symbol
The weighting function for the DTM/TDM (default is term-frequency, effectively unweighted)

Extra arguments passed on to `sparseMatrix`

Details

The arguments `term`, `document`, and `value` are passed by expression and support quasiquotation; you can unquote strings and symbols.

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

```r
## 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

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

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 'dictionary2'
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. The "bing" option comes from the included sentiments data frame, and others call the relevant function in the textdata package.

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

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

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)

## Not run:
get_sentiments("afinn")
get_sentiments("nrc")

## End(Not run)
get_sentiments("bing")

get_stopwords

Get a tidy data frame of a single stopword lexicon

Description

Get a specific stop word lexicon via the stopwords package’s stopwords function, in a tidy format with one word per row.

Usage

get_stopwords(language = "en", source = "snowball")

Arguments

language The language of the stopword lexicon specified as a two-letter ISO code, such as "es", "de", or "fr". Default is "en" for English. Use stopwords_getlanguages from stopwords to see available languages.

source The source of the stopword lexicon specified. Default is "snowball". Use stopwords_getsources from stopwords to see available sources.

Value

A tibble with two columns, word and lexicon. The parameter lexicon is "quanteda" in this case.
Examples

```r
library(dplyr)
get_stopwords()
get_stopwords(source = "smart")
get_stopwords("es", "snowball")
get_stopwords("ru", "snowball")
```

### Description

Tidy the results of a Latent Dirichlet Allocation or Correlated Topic Model.

### Usage

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

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

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

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

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

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

### Arguments

- `x` An LDA or CTM (or LDA_VEM/CTA_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 or CTM 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

```r
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 %>%
```
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

```r
## 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

```r
## 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)

---

**nma_words**

*NMA English negators, modals, and adverbs*

**Description**

English negators, modals, and adverbs, as a data frame. A few of these entries are two-word phrases instead of single words.

**Usage**

nma_words

**Format**

A data frame with 44 rows and 2 variables:

- **word** An English word or bigram
- **modifier** The modifier type for word, either "negator", "modal", or "adverb"

**Source**

http://saifmohammad.com/WebPages/SCL.html#NMA
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"

Details

Another dataset of English parts of speech, available only for non-commercial use, is available as part of SUBTLEXus at https://www.ugent.be/pp/experimentele-psychologie/en/research/documents/subtlexus/.

Source

https://archive.org/details/mobypartofspeech03203gut

Examples

```r
library(dplyr)

parts_of_speech

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

Reorder an x or y axis within facets

Description

Reorder a column before plotting with faceting, such that the values are ordered within each facet. This requires two functions: `reorder_within` applied to the column, then either `scale_x_reordered` or `scale_y_reordered` added to the plot. This is implemented as a bit of a hack: it appends `___` and then the facet at the end of each string.

Usage

```r
reorder_within(x, by, within, fun = mean, sep = "___", ...)  
scale_x_reordered(..., sep = "___")  
scale_y_reordered(..., sep = "___")
```

Arguments

- **x**: Vector to reorder.
- **by**: Vector of the same length, to use for reordering.
- **within**: Vector of the same length that will later be used for faceting.
- **fun**: Function to perform within each subset to determine the resulting ordering. By default, mean.
- **sep**: Separator to distinguish the two. You may want to set this manually if `___` can exist within one of your labels.
- **...**: In `reorder_within` arguments passed on to `reorder`. In the scale functions, extra arguments passed on to `scale_x_discrete` or `scale_y_discrete`.

Source

"Ordering categories within ggplot2 Facets" by Tyler Rinker: https://trinkerrstuff.wordpress.com/2016/12/23/ordering-categories-within-ggplot2-facets/

Examples

```r
library(tidyrr)  
library(ggplot2)  
iris_gathered <- gather(iris, metric, value, -Species)

# reordering doesn't work within each facet (see Sepal.Width):  
ggplot(iris_gathered, aes(reorder(Species, value), value)) +  
  geom_boxplot() +  
  facet_wrap(~ metric)
```
# reorder_within and scale_x_reordered work.
# (Note that you need to set scales = "free_x" in the facet)
ggplot(iris_gathered, aes(reorder_within(Species, value, metric), value)) +
  geom_boxplot() +
  scale_x_reordered() +
  facet_wrap(~ metric, scales = "free_x")

## sentiments

**Sentiment lexicon from Bing Liu and collaborators**

### Description

Lexicon for opinion and sentiment analysis in a tidy data frame. This dataset is included in this package with permission of the creators, and may be used in research, commercial, etc. contexts with attribution, using either the paper or URL below.

### Usage

```r
sentiments
```

### Format

A data frame with 6,786 rows and 2 variables:

- **word**: An English word
- **sentiment**: A sentiment for that word, either positive or negative.

### Details

This lexicon was first published in:


Words with non-ASCII characters were removed.

### Source

[https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html](https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html)
stm_tidiers Tidiers for Structural Topic Models from the stm package

Description

Tidy topic models fit by the stm package. The arguments and return values are similar to lda_tidiers.

Usage

```r
## S3 method for class 'STM'
tidy(x, matrix = c("beta", "gamma", "theta"),
     log = FALSE, document_names = NULL, ...)

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

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

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

Arguments

- **x**: An STM fitted model object from either `stm` or `estimateEffect` from the stm package.
- **matrix**: Whether to tidy the beta (per-term-per-topic, default) or gamma/theta (per-document-per-topic) matrix. The stm package calls this the theta matrix, but other topic modeling packages call this gamma.
- **log**: Whether beta/gamma/theta should be on a log scale, default FALSE
- **document_names**: Optional vector of document names for use with per-document-per-topic tidying
- **...**: Extra arguments, not used
- **data**: For `augment`, the data given to the stm function, either as a `dfm` from quanteda or as a tidied table with "document" and "term" columns

Value

- **tidy**: returns a tidied version of either the beta or gamma matrix if called on an object from `stm` or a tidied version of the estimated regressions if called on an object from `estimateEffect`.
- **augment**: must be provided a data argument, either a `dfm` from quanteda or a table 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 as a table with an additional column `.topic` with the topic assignment for that document-term combination.
- **glance**: always returns a one-row table, with columns
stm_tidiers

k  Number of topics in the model

docs Number of documents in the model

terms Number of terms in the model

iter Number of iterations used

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

See Also

lda_tidiers

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 structural topic model

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

topic Topic, as an integer
document Document name (if given in vector of document_names) or ID as an integer
gamma Probability of topic given document

If called on an object from estimateEffect, returns a table with columns

topic Topic, as an integer
term The term in the model being estimated and tested
estimate The estimated coefficient
std.error The standard error from the linear model
statistic t-statistic
p.value two-sided p-value

Examples

## Not run:
if (requireNamespace("stm", quietly = TRUE)) {
  library(dplyr)
  library(ggplot2)
  library(stm)
  library(janeaustenr)

  austen_sparse <- austen_books() %>%
    unnest_tokens(word, text) %>%
    anti_join(stop_words) %>%
    count(book, word) %>%
    cast_sparse(book, word, n)
  topic_model <- stm(austen_sparse, K = 12, verbose = FALSE, init.type = "Spectral")

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

# Examine the topics

td_beta %>%
group_by(topic) %>%
top_n(10, beta) %>%
ungroup() %>%
ggplot(aes(term, beta)) +
geom_col() +
facet_wrap(~ topic, scales = "free") +
coord_flip()

# tidy the document-topic combinations, with optional document names

td_gamma <- tidy(topic_model, matrix = "gamma",
document_names = rownames(austen_sparse))
td_gamma

# using stm's gardarianFit, we can tidy the result of a model
# estimated with covariates

effects <- estimateEffect(1:3 ~ treatment, gadarianFit, gadarian)
td_estimate <- tidy(effects)
td_estimate

}  # End(Not run)

---

stop_words  

Various lexicons for English stop words

Description

English stop words from three lexicons, as a data frame. The snowball 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"
**tdm_tidiers**  
*Tidy DocumentTermMatrix, TermDocumentMatrix, and related objects from the tm package*

**Source**
- http://snowball.tartarus.org/algorithms/english/stop.txt

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

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

## S3 method for class 'dfm'
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**
```r
if (requireNamespace("topicmodels", quietly = TRUE)) {
  data("AssociatedPress", package = "topicmodels")
  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

```r
## 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

```r
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),
```
tidytext

readerControl = list(reader = readReut21578XMLasPlain))

reuters
tidy(reuters)

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")

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. This function supports non-standard evaluation through the tidyeval framework.

Usage

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

Arguments

tbl  A data frame
output  Output column to be created as string or symbol.
input  Input column that gets split as string or symbol. The output/input arguments are passed by expression and support quasiquotation; you can unquote strings and symbols.
token  Unit for tokenizing, or a custom tokenizing function. Built-in options are "words" (default), "characters", "character_shingles", "ngrams", "skip_ngrams", "sentences", "lines", "paragraphs", "regex", "tweets" (tokenization by word that preserves usernames, hashtags, and URLs), and "ptb" (Penn Treebank). 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 convert tokens to lowercase. If tokens include URLs (such as with token = "tweets"), such converted URLs may no longer be correct.
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 tokenizers, such as strip_punct for "words" and "tweets", n and k for "ngrams" and "skip_ngrams", strip_url for "tweets", and 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 unless collapse = FALSE.

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

```r
library(dplyr)
library(janeaustenr)

d <- tibble(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(chapter, txt, token = "regex", pattern = "Chapter \d")
d %>%
  unnest_tokens(shingle, txt, token = "character_shingles", n = 4)

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

# tokenize HTML
h <- tibble(row = 1:2,
            text = c("<h1>Text <b>is</b> here</a>"))

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