05 Converting to and from non-tidy formats

H. David Shea

04 Aug 2021

Contents

Here we look at converting between tidy text format and other common formats used in text and natural language processing.

Tidying a document-term matrix

"One of the most common structures that text mining packages work with is the document-term matrix (or DTM). This is a matrix where:

- each row represents one document (such as a book or article),
- each column represents one term, and
- each value (typically) contains the number of appearances of that term in that document."

Tidying DocumentTermMatrix objects

```
data("AssociatedPress", package = "topicmodels")
AssociatedPress
#> <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
#> Non-/sparse entries: 302031/23220327
#> Sparsity           : 99%
#> Maximal term length: 18
#> Weighting          : term frequency (tf)

terms <- Terms(AssociatedPress)
head(terms)
#> [1] "aaron"      "abandon"    "abandoned"  "abandoning" "abbott"
#> [6] "abboud"

ap_td <- tidy(AssociatedPress)
ap_td %>%
    slice_sample(n = 10) %>%
    kable(caption = "DTM converted to tidy data frame with `tidy()`.")
```

Table 1: DTM converted to tidy data frame with `tidy()`.

document	term	count
1360	area	1
443	forcing	1
1129	popular	1
1893	transform	1
1300	missouri	1
1070	hard	1
1055	neighbor	1
1162	designed	1
1795	production	4
1852	think	1

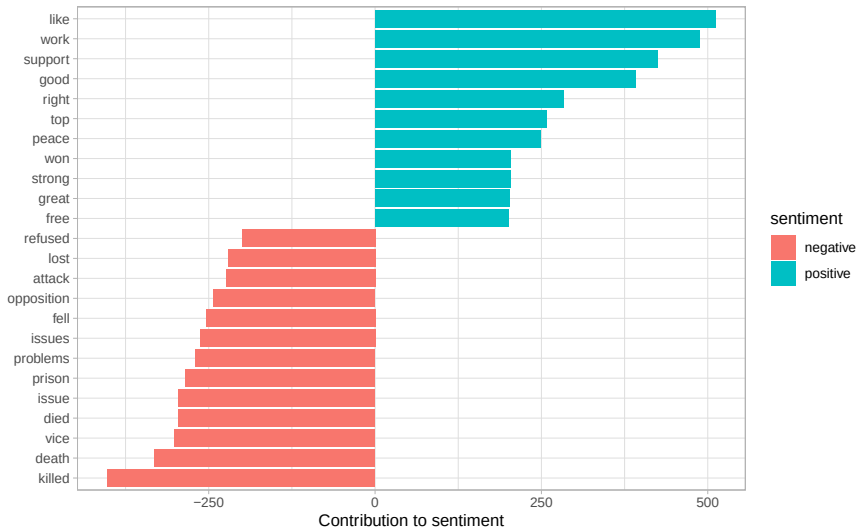
```
ap_sentiments <- ap_td %>%
    inner_join(get_sentiments("bing"), by = c("term" = "word"))

ap_sentiments %>%
    slice_sample(n = 10) %>%
    kable(caption = "Tidy formatted DTM joined with `bing` lexicon sentiment data")
```

Table 2: Tidy formatted DTM joined with `bing` lexicon sentiment data

document	term	count	sentiment
113	problems	1	negative
1147	threats	2	negative
590	dictator	2	negative
434	erode	1	negative
889	portable	1	positive
1457	conceded	1	negative
1986	jeopardy	1	negative
1783	insignificant	1	negative
170	tanks	2	negative
1114	inevitably	1	negative

```
ap_sentiments %>%
    count(sentiment, term, wt = count) %>%
    ungroup() %>%
    filter(n >= 200) %>%
    mutate(n = ifelse(sentiment == "negative",-n, n)) %>%
    mutate(term = reorder(term, n)) %>%
    ggplot(aes(n, term, fill = sentiment)) +
    geom_col() +
    labs(x = "Contribution to sentiment", y = NULL) +
    theme_light()
```



Tidying `dfm` objects

`dfm` (document-feature matrix) is another common format for text and natural langauge processing - specifically from the `quanteda` package.

```
data("data_corpus_inaugural", package = "quanteda")
inaug_dfm <- dfm(data_corpus_inaugural, verbose = FALSE)
inaug_dfm
#> Document-feature matrix of: 59 documents, 9,439 features (91.84% sparse) and 4 docvars.
#>                  features
#> docs              fellow-citizens  of the senate and house representatives :
#>   1789-Washington               1  71 116      1  48     2               2 1
#>   1793-Washington               0  11  13      0   2     0               0 1
#>   1797-Adams                    3 140 163      1 130     0               2 0
#>   1801-Jefferson                2 104 130      0  81     0               0 1
#>   1805-Jefferson                0 101 143      0  93     0               0 0
#>   1809-Madison                  1  69 104      0  43     0               0 0
#>                  features
#> docs              among vicissitudes
#>   1789-Washington     1            1
#>   1793-Washington     0            0
#>   1797-Adams          4            0
#>   1801-Jefferson      1            0
#>   1805-Jefferson      7            0
#>   1809-Madison        0            0
#> [ reached max_ndoc ... 53 more documents, reached max_nfeat ... 9,429 more features ]

inaug_td <- tidy(inaug_dfm)
inaug_td %>%
    slice_sample(n = 10) %>%
    kable(caption = "`dfm` object containing US President inagural speechees converted to tidy format w
```

Table 3: `dfm` object containing US President inagural speechees converted to tidy format with `tidy()`.

document	term	count
1873-Grant	they	2
1845-Polk	position	1
1829-Jackson	united	1
1933-Roosevelt	measures	4
1845-Polk	when	4
2009-Obama	like	2
1869-Grant	metals	1
1845-Polk	reverse	1
1917-Wilson	neither	2
1853-Pierce	wherever	1

```
inaug_tf_idf <- inaug_td %>%
    bind_tf_idf(term, document, count) %>%
    arrange(desc(tf_idf))

inaug_tf_idf %>%
    slice_max(tf_idf, n = 10) %>%
    kable(caption = "Highest `tf-idf` values for terms by each US Presidential inauguration speech.")
```

Table 4: Highest `tf-idf` values for terms by each US Presidential inauguration speech.

document	term	count	tf	idf	tf_idf
1793-Washington	arrive	1	0.0068027	4.077537	0.0277383
1793-Washington	upbraidings	1	0.0068027	4.077537	0.0277383
1793-Washington	violated	1	0.0068027	3.384390	0.0230231
1793-Washington	willingly	1	0.0068027	3.384390	0.0230231
1793-Washington	incurring	1	0.0068027	3.384390	0.0230231
1793-Washington	previous	1	0.0068027	2.978925	0.0202648
1793-Washington	knowingly	1	0.0068027	2.978925	0.0202648
1793-Washington	injunctions	1	0.0068027	2.978925	0.0202648
1793-Washington	witnesses	1	0.0068027	2.978925	0.0202648
1793-Washington	besides	1	0.0068027	2.691243	0.0183078

```
speeches <- c("1933-Roosevelt",
              "1861-Lincoln",
              "1961-Kennedy",
              "2009-Obama")

inaug_tf_idf %>%
    filter(document %in% speeches) %>%
    group_by(document) %>%
    slice_max(tf_idf, n = 10, with_ties = FALSE) %>%
    ungroup() %>%
    mutate(term = reorder_within(term, tf_idf, document)) %>%
    ggplot(aes(term, tf_idf, fill = document)) +
```

```
    geom_col(show.legend = FALSE) +
    facet_wrap( ~ document, scales = "free") +
    coord_flip() +
    scale_x_reordered() +
    labs(x = NULL,
         y = "tf-idf") +
    theme_light()
```

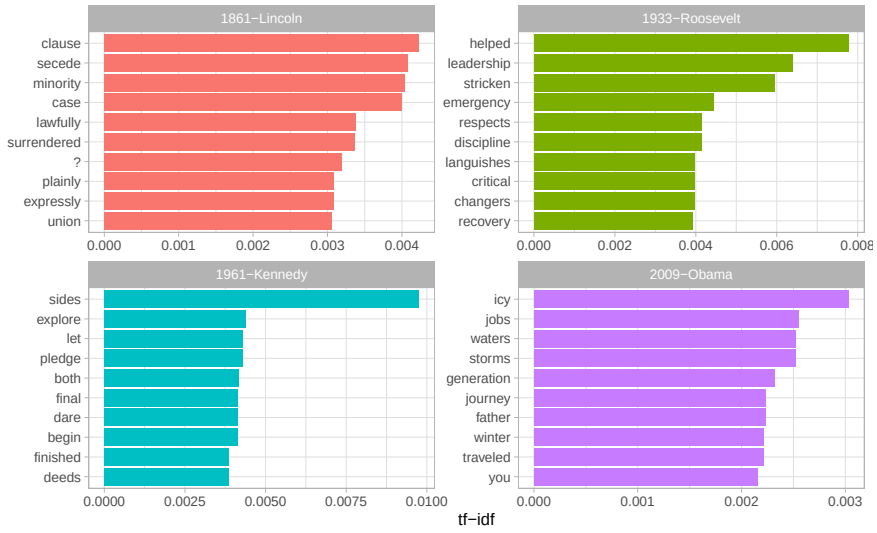


Figure 1: The terms with the highest ‘tf-idf‘ from each of four selected inaugural addresses.

```
year_term_counts <- inaug_td %>%
    extract(document, "year", "(\\d+)", convert = TRUE) %>%
    complete(year, term, fill = list(count = 0)) %>%
    group_by(year) %>%
    mutate(year_total = sum(count))

year_term_counts %>%
    filter(term %in% c("god", "america", "foreign", "union", "constitution", "freedom")) %>%
    ggplot(aes(year, count / year_total)) +
    geom_point() +
    geom_smooth() +
    facet_wrap( ~ term, scales = "free_y") +
    scale_y_continuous(labels = scales::percent_format()) +
    labs(y = "% frequency of word in inaugural address") +
    theme_light()
```

“These examples show how you can use tidytext, and the related suite of tidy tools, to analyze sources even if their origin was not in a tidy format.”

Casting tidy text data into a matrix

There are three verbs provided in `tidytext` for converting from tidy format to the alternative formats discussed.

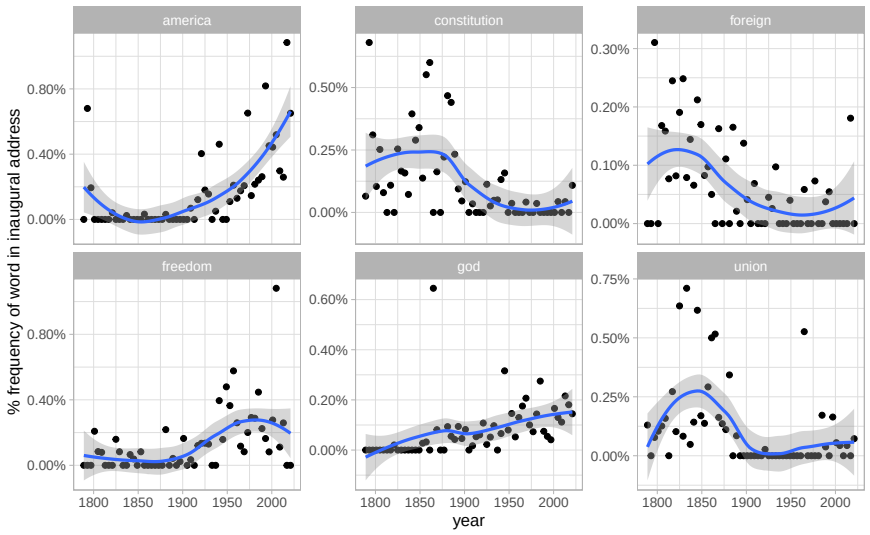


Figure 2: Changes in word frequency over time within Presidential inaugural addresses, for six selected terms

- `tidy_dtm()` to `dtm` format
- `tidy_dfm()` to `dfm` format
- `tidy_sparse()` to sparse matrix format

```
ap_td %>%
    cast_dtm(document, term, count)
#> <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
#> Non-/sparse entries: 302031/23220327
#> Sparsity           : 99%
#> Maximal term length: 18
#> Weighting          : term frequency (tf)

ap_td %>%
    cast_dfm(document, term, count)
#> Document-feature matrix of: 2,246 documents, 10,473 features (98.72% sparse) and 0 docvars.
#>     features
#> docs adding adult ago alcohol allegedly allen apparently appeared arrested
#>    1      1     2   1       1         1     1          2        1        1
#>    2      0     0   0       0         0     0          0        1        0
#>    3      0     0   1       0         0     0          0        1        0
#>    4      0     0   3       0         0     0          0        0        0
#>    5      0     0   0       0         0     0          0        0        0
#>    6      0     0   2       0         0     0          0        0        0
#>     features
#> docs assault
#>    1       1
#>    2       0
#>    3       0
#>    4       0
#>    5       0
#>    6       0
#> [ reached max_ndoc ... 2,240 more documents, reached max_nfeat ... 10,463 more features ]
```

```
m <- ap_td %>%
    cast_sparse(document, term, count)

class(m)
#> [1] "dgCMatrix"
#> attr(,"package")
#> [1] "Matrix"
dim(m)
#> [1]  2246 10473

austen_dtm <- austen_books() %>%
    unnest_tokens(word, text) %>%
    count(book, word) %>%
    cast_dtm(book, word, n)

austen_dtm
#> <<DocumentTermMatrix (documents: 6, terms: 14520)>>
#> Non-/sparse entries: 40379/46741
#> Sparsity           : 54%
#> Maximal term length: 19
#> Weighting          : term frequency (tf)
```

Tidying corpus objects with metadata

Another common format for text is called a "corpus". These store metadata along with text.

```
data("acq")
acq
#> <<VCorpus>>
#> Metadata:  corpus specific: 0, document level (indexed): 0
#> Content:  documents: 50

acq[[1]]
#> <<PlainTextDocument>>
#> Metadata:  15
#> Content:  chars: 1287

acq_td <- tidy(acq)
acq_td
#> # A tibble: 50 x 16
#>    author  datetimestamp       description heading  id    language origin topics
#>    <chr>   <dttm>              <chr>       <chr>    <chr> <chr>    <chr>  <chr>
#>  1 <NA>    1987-02-26 15:18:06 ""          COMPUTE~ 10    en       Reute~ YES
#>  2 <NA>    1987-02-26 15:19:15 ""          OHIO MA~ 12    en       Reute~ YES
#>  3 <NA>    1987-02-26 15:49:56 ""          MCLEAN'~ 44    en       Reute~ YES
#>  4 By Cal~ 1987-02-26 15:51:17 ""          CHEMLAW~ 45    en       Reute~ YES
#>  5 <NA>    1987-02-26 16:08:33 ""          <COFAB ~ 68    en       Reute~ YES
#>  6 <NA>    1987-02-26 16:32:37 ""          INVESTM~ 96    en       Reute~ YES
#>  7 By Pat~ 1987-02-26 16:43:13 ""          AMERICA~ 110   en       Reute~ YES
#>  8 <NA>    1987-02-26 16:59:25 ""          HONG KO~ 125   en       Reute~ YES
#>  9 <NA>    1987-02-26 17:01:28 ""          LIEBERT~ 128   en       Reute~ YES
#> 10 <NA>    1987-02-26 17:08:27 ""          GULF AP~ 134   en       Reute~ YES
```

```
#> # ... with 40 more rows, and 8 more variables: lewissplit <chr>,
#> #   cgisplit <chr>, oldid <chr>, places <named list>, people <lgl>, orgs <lgl>,
#> #   exchanges <lgl>, text <chr>

acq_tokens <- acq_td %>%
    select(-places) %>%
    unnest_tokens(word, text) %>%
    anti_join(stop_words, by = "word")

acq_tokens %>%
    count(word, sort = TRUE) %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Most common words in sampled Reuters articles.")
```

Table 5: Most common words in sampled Reuters articles.

word	n
dlrs	100
pct	70
mln	65
company	63
shares	52
reuter	50
stock	46
offer	34
share	34
american	28

```
acq_tokens %>%
    count(id, word) %>%
    bind_tf_idf(word, id, n) %>%
    arrange(desc(tf_idf)) %>%
    slice_max(tf_idf, n = 10) %>%
    kable(caption = "Highest tf-idf value words in sampled Reuters articles.")
```

Table 6: Highest tf-idf value words in sampled Reuters articles.

id	word	n	tf	idf	tf_idf
186	groupe	2	0.1333333	3.912023	0.5216031
128	liebert	3	0.1304348	3.912023	0.5102639
474	esselte	5	0.1086957	3.912023	0.4252199
371	burdett	6	0.1034483	3.912023	0.4046920
442	hazleton	4	0.1025641	3.912023	0.4012331
199	circuit	5	0.1020408	3.912023	0.3991860
162	suffield	2	0.1000000	3.912023	0.3912023
498	west	3	0.1000000	3.912023	0.3912023
441	rmj	8	0.1212121	3.218876	0.3901668
467	nursery	3	0.0967742	3.912023	0.3785829

Example: mining financial articles

Some JAVA requirement for `tm.plugin.webmining` is not yet implemented for Mac M1 machines (and causes the library loading to hang), so this section will need to await that implementation.