

## CHAPTER 8

# Case Study: Mining NASA Metadata

There are over 32,000 datasets hosted and/or maintained by NASA; these datasets cover topics from Earth science to aerospace engineering to management of NASA itself. We can use the metadata for these datasets to understand the connections between them.



What is metadata? Metadata is a term that refers to data that gives information about other data; in this case, the metadata informs users about what is in these numerous NASA datasets but does not include the content of the datasets themselves.

eg, you can use XML tags to define metadata in a transcript of a medieval text  
(Look at Medieval Mary Magdalene project)

The metadata includes information like the title of the dataset, a description field, what organization(s) within NASA is responsible for the dataset, keywords for the dataset that have been assigned by a human being, and so forth. NASA places a high priority on making its data open and accessible, even requiring all NASA-funded research to be openly accessible online. The metadata for all its datasets is publicly available online in JSON format.

JavaScript object Notation (like we learned in computational Thinking)

In this chapter, we will treat the NASA metadata as a text dataset and show how to implement several tidy text approaches with this real-life text. We will use word co-occurrences and correlations, tf-idf, and topic modeling to explore the connections between the datasets. Can we find datasets that are related to each other? Can we find clusters of similar datasets? Since we have several text fields in the NASA metadata, most importantly the title, description, and keyword fields, we can explore the connections between the fields to better understand the complex world of data at NASA. This type of approach can be extended to any domain that deals with text, so let's take a look at this metadata and get started.

your RQ and arguments

# How Data Is Organized at NASA

First, let's download the JSON file and take a look at the names of what is stored in the metadata.

```
library(jsonlite)
metadata <- fromJSON("https://data.nasa.gov/data.json")
names(metadata$dataset)
```

i.e. these  
are all of  
the categories  
of metadata  
that we can  
sort it by.

## [1] "_id"	"@type"	"accessLevel"
## [4] "accrualPeriodicity"	"bureauCode"	"contactPoint"
## [7] "description"	"distribution"	"identifier"
## [10] "issued"	"keyword"	"landingPage"
## [13] "language"	"modified"	"programCode"
## [16] "publisher"	"spatial"	"temporal"
## [19] "theme"	"title"	"license"
## [22] "isPartOf"	"references"	"rights"
## [25] "describedBy"		

We see here that we could extract information from who publishes each dataset to what license each dataset is released under.

It seems likely that the title, description, and keywords for each dataset may be most fruitful for drawing connections between datasets. Let's check them out.

I like this  
case study's  
methodology.  
How can I  
find other  
metadata  
JSON files  
on my own?

```
class(metadata$dataset$title)
## [1] "character"
class(metadata$dataset$description)
## [1] "character"
class(metadata$dataset$keyword)
## [1] "list"
```

variable types

These 3 functions on the left take  
a few mins to run, so I must be  
patient & let them finish before  
going on.

## Wrangling and Tidying the Data

Let's set up separate tidy data frames for title, description, and keyword, keeping the dataset IDs for each so that we can connect them later in the analysis if necessary.

use "tibble" library(dplyr)  
because the  
previous  
command  
("data\_frame")  
is deprecated. tibble  
nasa\_title <- data\_frame(id = metadata\$  
title = metadata\$  
nasa\_title  
## # A tibble: 32,089 × 2  
## id <chr>  
## 1 55942a57c63a7fe59b495a77

I am missing an "ID # column: I  
only have the "title" column. I  
suspect that the regex we used to  
import the "id" column isn't right  
anymore, so I tried to check the  
JSON to see how I could adjust the  
regex. But the JSON took too long to  
load... how can I figure out what  
went wrong when loading in IDs?  
Wait, JSON didn't load... I looked for the  
"ID # "55942a..." and it didn't exist.  
What?!

```

## 2 55942a57c63a7fe59b495a78      15 Minute Stream Flow Data: USGS (FIFE
## 3 55942a58c63a7fe59b495a79      15 Minute Stream Flow Data: USGS (FIFE
## 4 55942a58c63a7fe59b495a7a 2000 Pilot Environmental Sustainability Index (ESI
## 5 55942a58c63a7fe59b495a7b 2000 Pilot Environmental Sustainability Index (ESI
## 6 55942a58c63a7fe59b495a7c 2000 Pilot Environmental Sustainability Index (ESI
## 7 55942a58c63a7fe59b495a7d      2001 Environmental Sustainability Index (ESI
## 8 55942a58c63a7fe59b495a7e      2001 Environmental Sustainability Index (ESI
## 9 55942a58c63a7fe59b495a7f      2001 Environmental Sustainability Index (ESI
## 10 55942a58c63a7fe59b495a80     2001 Environmental Sustainability Index (ESI
## # ... with 32,079 more rows

```

These are just a few example titles from the datasets we will be exploring. Notice that we have the NASA-assigned IDs here, and also that there are duplicate titles on separate datasets.

```

nasa_desc <- data_frame(id = metadata$dataset$id $oid, missing ID column!
                           desc = metadata$dataset$description) ✓

nasa_desc %>%
  select(desc) %>% ✓ gives me a 5x1, which is
  sample_n(5) ✓ good (as it should)

## # A tibble: 5 × 1
##   ✓
##   ✓
## 1 MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument
## 2 Fatigue Countermeasures: A Meta-Analysis
## 3 Mobile communications systems require programmable embedded platforms that
## 4 The Doppler Aerosol Wind (DAWN), a pulsed lidar, operated aboard a NASA DC-8
## 5 MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument

```

Here we see the first part of several selected description fields from the metadata.

Now we can build the tidy data frame for the keywords. For this one, we need to use `unnest()` from `tidyverse`, because they are in a list-column.

```

library(tidyverse)

nasa_keyword <- data_frame(id = metadata$dataset$id $oid, missing ID column!
                            keyword = metadata$dataset$keyword) %>%
  unnest(keyword) ✓

nasa_keyword

## # A tibble: 126,814 × 2
##       id      keyword
##   <chr>    <chr>
## 1 55942a57c63a7fe59b495a77 EARTH SCIENCE
## 2 55942a57c63a7fe59b495a77 HYDROSPHERE
## 3 55942a57c63a7fe59b495a77 SURFACE WATER
## 4 55942a57c63a7fe59b495a78 EARTH SCIENCE
## 5 55942a57c63a7fe59b495a78 HYDROSPHERE
## 6 55942a57c63a7fe59b495a78 SURFACE WATER
## 7 55942a58c63a7fe59b495a79 EARTH SCIENCE

```

My output, as expected, does not have an id column, but it DOES have a keyword column. The words there are, from T to B, "earth," "unknown," "international rosetta mission," v.v.. ✓

```

## 8 55942a58c63a7fe59b495a79 HYDROSPHERE
## 9 55942a58c63a7fe59b495a79 SURFACE WATER
## 10 55942a58c63a7fe59b495a7a EARTH SCIENCE
## # ... with 126,804 more rows

```

This is a tidy data frame because we have one row for each keyword; this means we will have multiple rows for each dataset because a dataset can have more than one keyword.

*it turns "international rosetta mission" into 3 separate rows*  
Now it is time to use tidytext's `unnest_tokens()` for the title and description fields so we can do the text analysis. Let's also remove stop words from the titles and descriptions. We will not remove stop words from the keywords, because those are short, human-assigned keywords like "RADIATION" or "CLIMATE INDICATORS."

```

library(tidytext) ✓
nasa_title <- nasa_title %>% ✓
  unnest_tokens(word, title) %>%
  anti_join(stop_words)
?Another error: "error: must
extract column with a single valid
subscript." (Line 70-72).
nasa_desc <- nasa_desc %>%
  unnest_tokens(word, desc) %>%
  anti_join(stop_words) ✓ This one runs fine - I can tidy
all of the desc into tokens

```

These are now in the tidy text format that we have been working with throughout this book, with one token (word, in this case) per row; let's take a look before we move on in our analysis.

```

nasa_title
## # A tibble: 210,914 × 2
##       id      word
##   <chr>    <chr>
## 1 56d07ee5a759fdadc44e5923 marble
## 2 56d07ee5a759fdadc44e5923 epic
## 3 56d07c16a759fdadc44e5922 fitara
## 4 56d07c16a759fdadc44e5922 ocio
## 5 56cf5b00a759fdadc44e5849 implementing
## 6 56cf5b00a759fdadc44e5846 receding
## 7 56cf5b00a759fdadc44e5846 recursive
## 8 56cf5b00a759fdadc44e5840 complaints
## 9 56cf5b00a759fdadc44e583b score
## 10 56cf5b00a759fdadc44e583a fix
## # ... with 210,904 more rows

nasa_desc
## # A tibble: 2,677,811 × 2
##       id      word
##   <chr>    <chr>
## 1 56d07c16a759fdadc44e5922 fitara
## 2 56d07c16a759fdadc44e5922 ocio
## 3 56cf5b00a759fdadc44e584a degradation's

```

```

## 4 56cf5b00a759fdadc44e5847      dchwp1s
## 5 56cf5b00a759fdadc44e5847      dchwp1sp
## 6 56cf5b00a759fdadc44e5847      dchwdp
## 7 56cf5b00a759fdadc44e5847      dchwsnf
## 8 56cf5b00a759fdadc44e5847      dchwssf
## 9 56cf5b00a759fdadc44e5847      bursting
## 10 56cf5b00a759fdadc44e5847 consequentially
## # ... with 2,677,801 more rows

```

## Some Initial Simple Exploration

What are the most common words in the NASA dataset titles? We can use `count()` from `dplyr` to check this out.

```

nasa_title %>%
  count(word, sort = TRUE)

## # A tibble: 11,614 × 2
##       word     n
##       <chr> <int>
## 1 project    7735
## 2 data        3354
## 3 level       2841
## 4 global      2400
## 5 v1          1809
## 6 daily       1478
## 7 aura         1397
## 8 v3          1364
## 9 aura         1363
## 10 v2          1311
## # ... with 11,604 more rows

```

I wonder if there's a way to filter these out if we don't really consider them "words"

What about the descriptions?

```

nasa_desc %>%
  count(word, sort = TRUE)

## # A tibble: 35,940 × 2
##       word     n
##       <chr> <int>
## 1 data       68871
## 2 modis      24420
## 3 global     23028
## 4 2          16599
## 5 1          15770
## 6 system     15480
## 7 product    14780
## 8 aqua       14738
## 9 earth       14373
## 10 resolution 13879
## # ... with 35,930 more rows

```

Words like “data” and “global” are used very often in NASA titles and descriptions. We may want to remove digits and some “words” like “v1” from these data frames for many types of analyses; they are not too meaningful for most audiences.



We can do this by making a list of custom stop words and using `anti_join()` to remove them from the data frame, just like we removed the default stop words that are in the tidytext package. This approach can be used in many instances and is a great tool to bear in mind.

```
my_stopwords <- data_frame(word = c(as.character(1:10),  
                                "v1", "v03", "l2", "l3", "l4", "v5.2.0",  
                                "v003", "v004", "v005", "v006", "v7"))  
  
nasa_title <- nasa_title %>%  
  anti_join(my_stopwords)  
nasa_desc <- nasa_desc %>%  
  anti_join(my_stopwords)
```

What are the most common keywords?

```
nasa_keyword %>%  
  group_by(keyword) %>%  
  count(sort = TRUE)  
  
## # A tibble: 1,774 × 2  
##   keyword      n  
##   <chr>     <int>  
## 1 EARTH SCIENCE 14362 ← I wonder why these  
## 2 Project       7452 words (plural) count as  
## 3 ATMOSPHERE    7321 one keyword; is it like  
## 4 Ocean Color   7268 a comma separated  
## 5 Ocean Optics   7268 value?  
## 6 Oceans         7268  
## 7 completed      6452  
## 8 ATMOSPHERIC WATER VAPOR 3142  
## 9 OCEANS          2765  
## 10 LAND SURFACE   2720  
## # ... with 1,764 more rows
```

We likely want to change all of the keywords to either lower- or uppercase to get rid of duplicates like “OCEANS” and “Oceans.” Let’s do that here.

```
nasa_keyword <- nasa_keyword %>%  
  mutate(keyword = toupper(keyword))
```

## Word Co-occurrences and Correlations

As a next step, let’s examine which words commonly occur together in the titles, descriptions, and keywords of NASA datasets, as described in Chapter 4. We can then

examine word networks for these fields; this may help us see, for example, which datasets are related to each other.

## Networks of Description and Title Words

We can use `pairwise_count()` from the `widyr` package to count how many times each pair of words occurs together in a title or description field.

```
library(widyr)

title_word_pairs <- nasa_title %>%
  pairwise_count(word, id, sort = TRUE, upper = FALSE)

title_word_pairs

## # A tibble: 156,689 × 3
##       item1     item2     n
##       <chr>     <chr> <dbl>
## 1 system project  796
## 2 lba      eco      683
## 3 airs     aqua     641
## 4 level    aqua     623
## 5 level    airs      612
## 6 aura     omi      607
## 7 global   grid      597
## 8 global   daily     574
## 9 data     boreas    551
## 10 ground   gpm      550
## # ... with 156,679 more rows
```

These are the pairs of words that occur together most often in title fields. Some of these words are obviously acronyms used within NASA, and we see how often words like “project” and “system” are used.

```
desc_word_pairs <- nasa_desc %>%
  pairwise_count(word, id, sort = TRUE, upper = FALSE)

desc_word_pairs

## # A tibble: 10,889,084 × 3
##       item1     item2     n
##       <chr>     <chr> <dbl>
## 1 data      global  9864
## 2 data      resolution 9302
## 3 instrument resolution 8189
## 4 data      surface  8180
## 5 global    resolution 8139
## 6 data      instrument 7994
## 7 data      system   7870
## 8 resolution bands   7584
## 9 data      earth    7576
```

```
## 10      orbit resolution 7462  
## # ... with 10,889,074 more rows
```

These are the pairs of words that occur together most often in descriptor fields. “Data” is a very common word in description fields; there is no shortage of data in the datasets at NASA!

Let's plot networks of these co-occurring words so we can see these relationships better in Figure 8-1. We will again use the `ggraph` package for visualizing our networks.

```
library(ggplot2)
library(igraph)
library(ggraph)

set.seed(1234)
title_word_pairs %>%
  filter(n >= 250) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                point.padding = unit(0.2, "lines")) +
  theme_void()
```

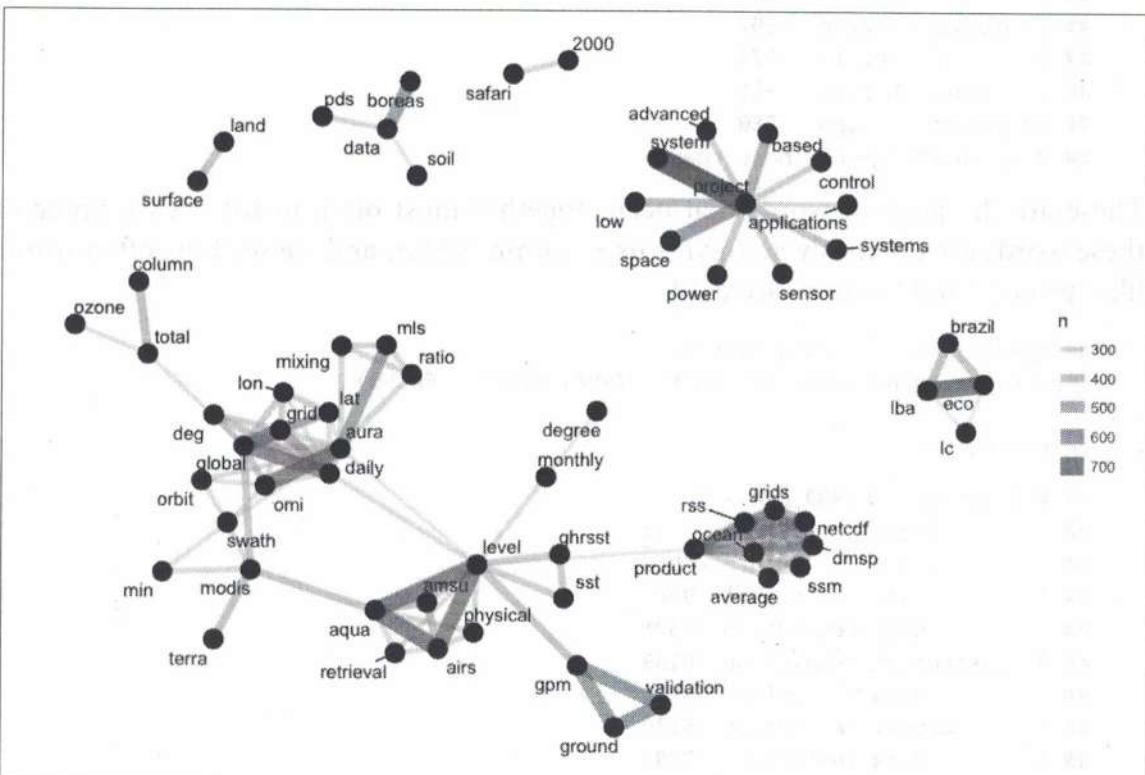


Figure 8-1. Word network in NASA dataset titles

```
## 10    orbit resolution 7462
## # ... with 10,889,074 more rows
```

These are the pairs of words that occur together most often in descriptor fields. “Data” is a very common word in description fields; there is no shortage of data in the datasets at NASA!

Let’s plot networks of these co-occurring words so we can see these relationships better in Figure 8-1. We will again use the ggraph package for visualizing our networks.

```
library(ggplot2)
library(igraph)
library(ggraph)

set.seed(1234)
title_word_pairs %>%
  filter(n >= 250) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```

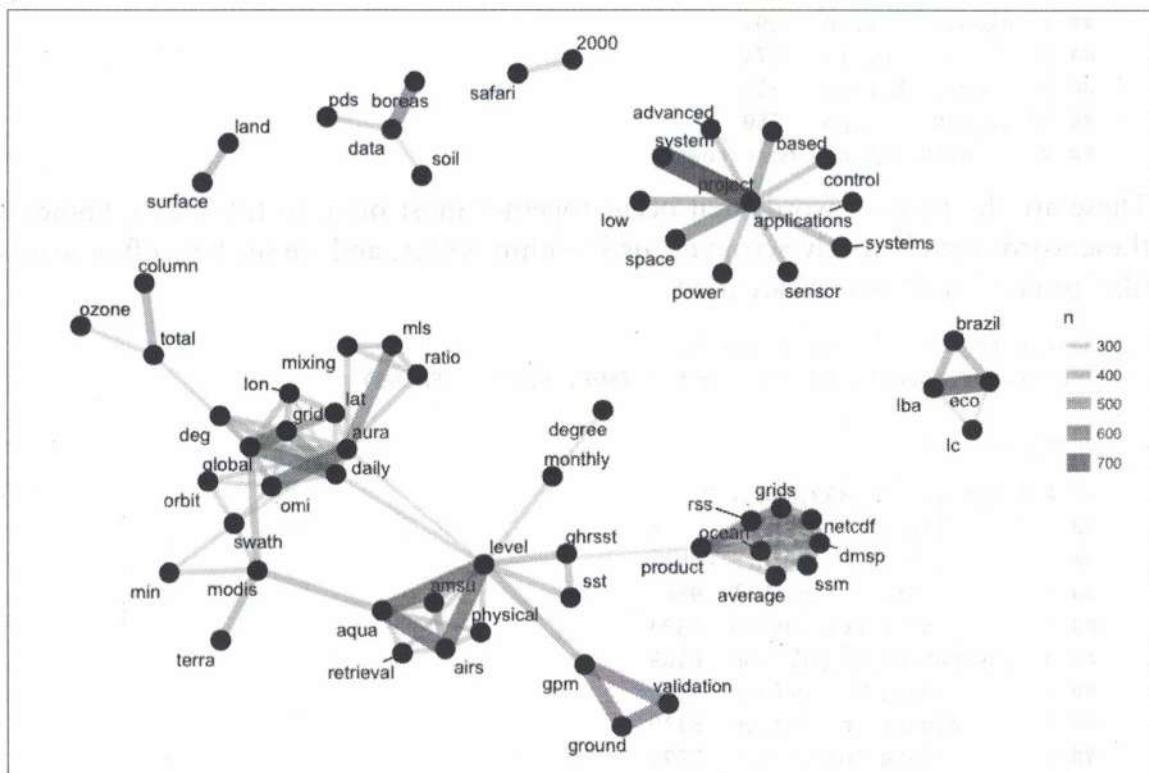


Figure 8-1. Word network in NASA dataset titles

We see some clear clustering in this network of title words; words in NASA dataset titles are largely organized into several families of words that tend to go together.

What about the words from the description fields (Figure 8-2)?

```
set.seed(1234)
desc_word_pairs %>%
  filter(n >= 5000) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "darkred") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                point.padding = unit(0.2, "lines")) +
  theme_void()
```

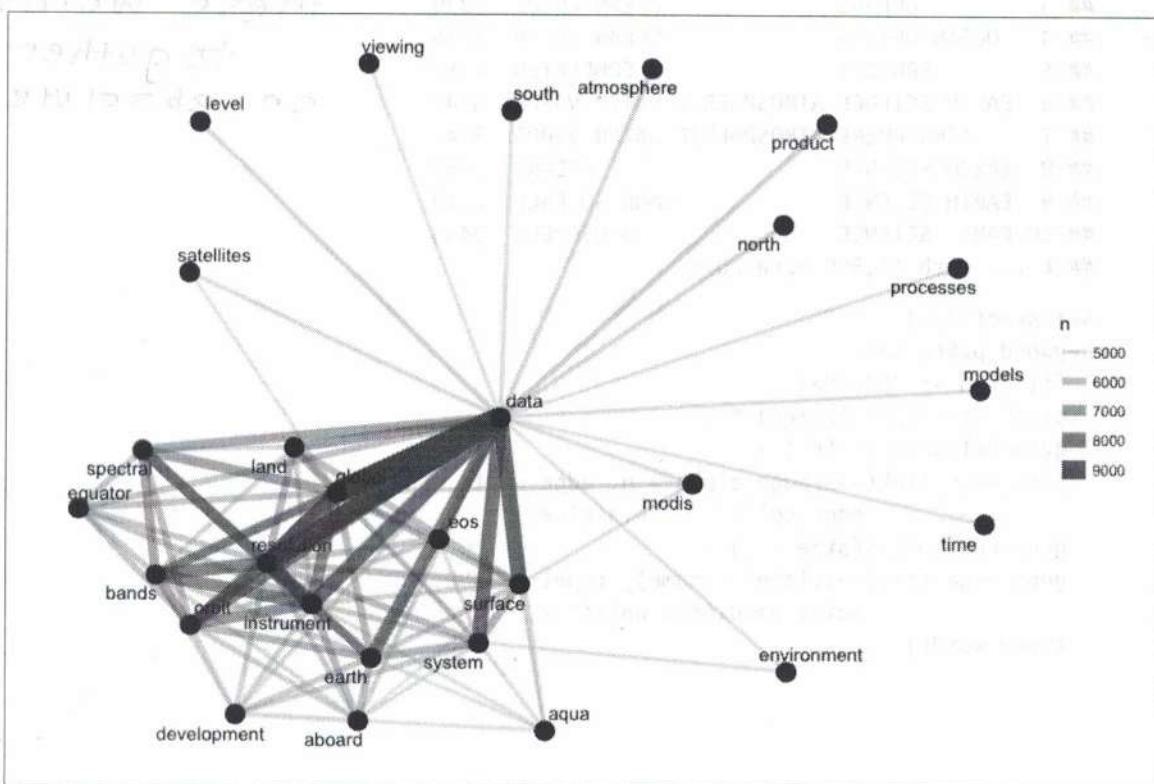


Figure 8-2. Word network in NASA dataset descriptions

Figure 8-2 shows such *strong* connections between the top dozen or so words (words like “data,” “global,” “resolution,” and “instrument”) that we do not see a clear clustering structure in the network. We may want to use tf-idf (as described in detail in Chapter 3) as a metric to find characteristic words for each description field, instead of looking at counts of words.

so words (words  
in a clear cluster-  
ed in detail in  
a field, instead  
I don't  
understand  
this  
explanation

Since there is  
a strong  
connection, isn't  
there a clear  
clustering network?

## Networks of Keywords

Next, let's make a network of the keywords in Figure 8-3 to see which keywords commonly occur together in the same datasets.

```
keyword_pairs <- nasa_keyword %>%
  pairwise_count(keyword, id, sort = TRUE, upper = FALSE)

keyword_pairs
```

“fruit” & “fly”  
commonly  
occur together

“how many times  
these words occur  
together as  
an absolute value”

item1	item2	n
OCEANS	OCEAN OPTICS	7324
EARTH SCIENCE	ATMOSPHERE	7318
OCEANS	OCEAN COLOR	7270
OCEAN OPTICS	OCEAN COLOR	7270
PROJECT	COMPLETED	6450
EARTH SCIENCE	ATMOSPHERIC WATER VAPOR	3142
ATMOSPHERE	ATMOSPHERIC WATER VAPOR	3142
EARTH SCIENCE	OCEANS	2762
EARTH SCIENCE	LAND SURFACE	2718
EARTH SCIENCE	BIOSPHERE	2448

## # ... with 13,380 more rows

```
set.seed(1234)
keyword_pairs %>%
  filter(n >= 700) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n),
                 edge_colour = "royalblue") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                point.padding = unit(0.2, "lines")) +
  theme_void()
```

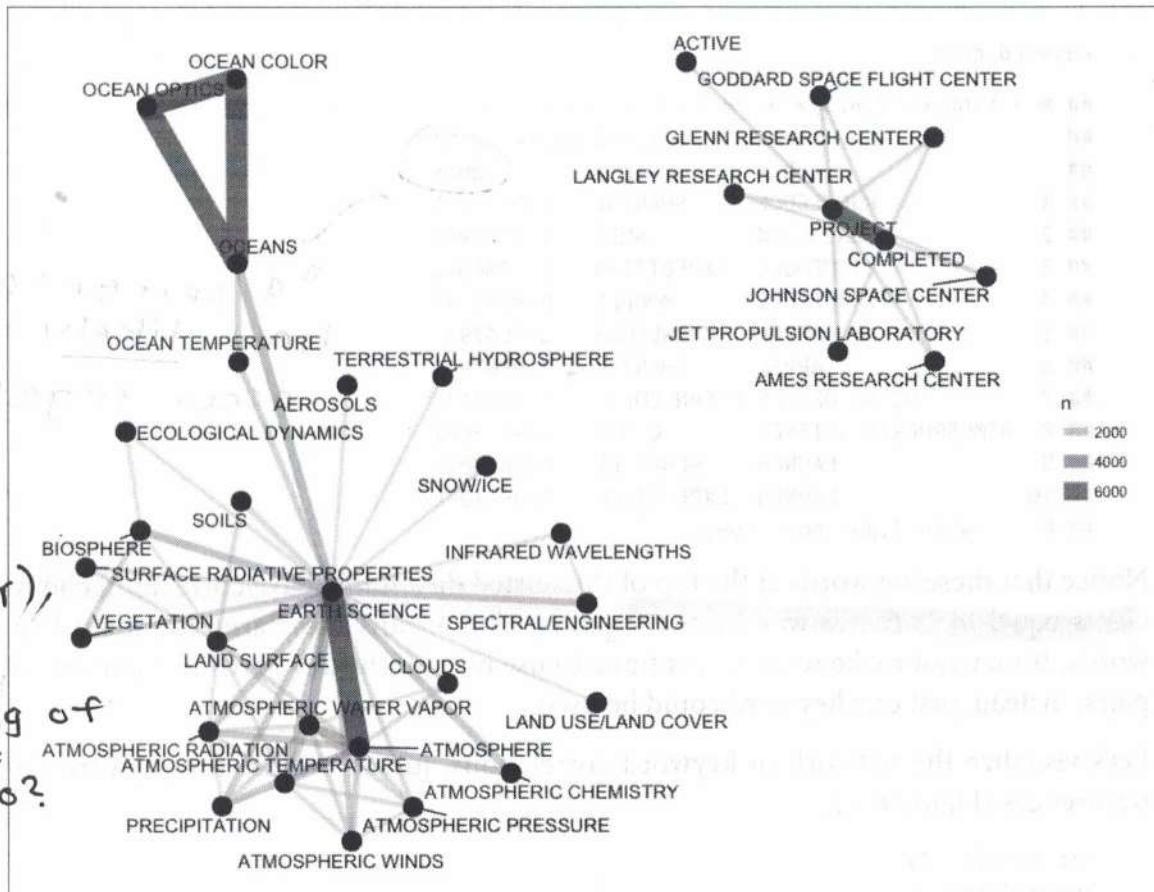


Figure 8-3. Co-occurrence network in NASA dataset keywords

We definitely see clustering here, and strong connections between keywords like “OCEANS,” “OCEAN OPTICS,” and “OCEAN COLOR,” or “PROJECT” and “COMPLETED.”



These are the most commonly co-occurring words, but also just the most common keywords in general.

To examine the relationships among keywords in a different way, we can find the correlation among the keywords as described in Chapter 4. This looks for those keywords that are more likely to occur together than with other keywords in a description field.

```
keyword_cors <- nasa_keyword %>%
  group_by(keyword) %>%
  filter(n() >= 50) %>%
  pairwise_cor(keyword, id, sort = TRUE, upper = FALSE)
```

“fruit” and “fly”

“fruit” Word Co-occurrences and Correlations | 135  
and “paper”

```

keyword_cors

## # A tibble: 7,875 × 3
##   item1     item2 correlation
##   <chr>     <chr>    <dbl>
## 1 KNOWLEDGE SHARING  1.0000000
## 2 DASHLINK  AMES     1.0000000
## 3 SCHEDULE  EXPEDITION 1.0000000
## 4 TURBULENCE MODELS   0.9971871
## 5 APPEL      KNOWLEDGE 0.9967945
## 6 APPEL      SHARING   0.9967945
## 7 OCEAN OPTICS OCEAN COLOR  0.9952123
## 8 ATMOSPHERIC SCIENCE CLOUD   0.9938681
## 9 LAUNCH     SCHEDULE  0.9837078
## 10 LAUNCH    EXPEDITION 0.9837078
## # ... with 7,865 more rows

```

» a percentage of how likely these occur together

Notice that these keywords at the top of this sorted data frame have correlation coefficients equal to 1; they always occur together. This means these are redundant keywords. It may not make sense to continue to use both of the keywords in these sets of pairs; instead, just one keyword could be used.

Let's visualize the network of keyword correlations, just as we did for keyword co-occurrences (Figure 8-4).

```

set.seed(1234)
keyword_cors %>%
  filter(correlation > .6) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation),
                 edge_colour = "royalblue") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                point.padding = unit(0.2, "lines")) +
  theme_void()

```

so useful  
to see  
this!

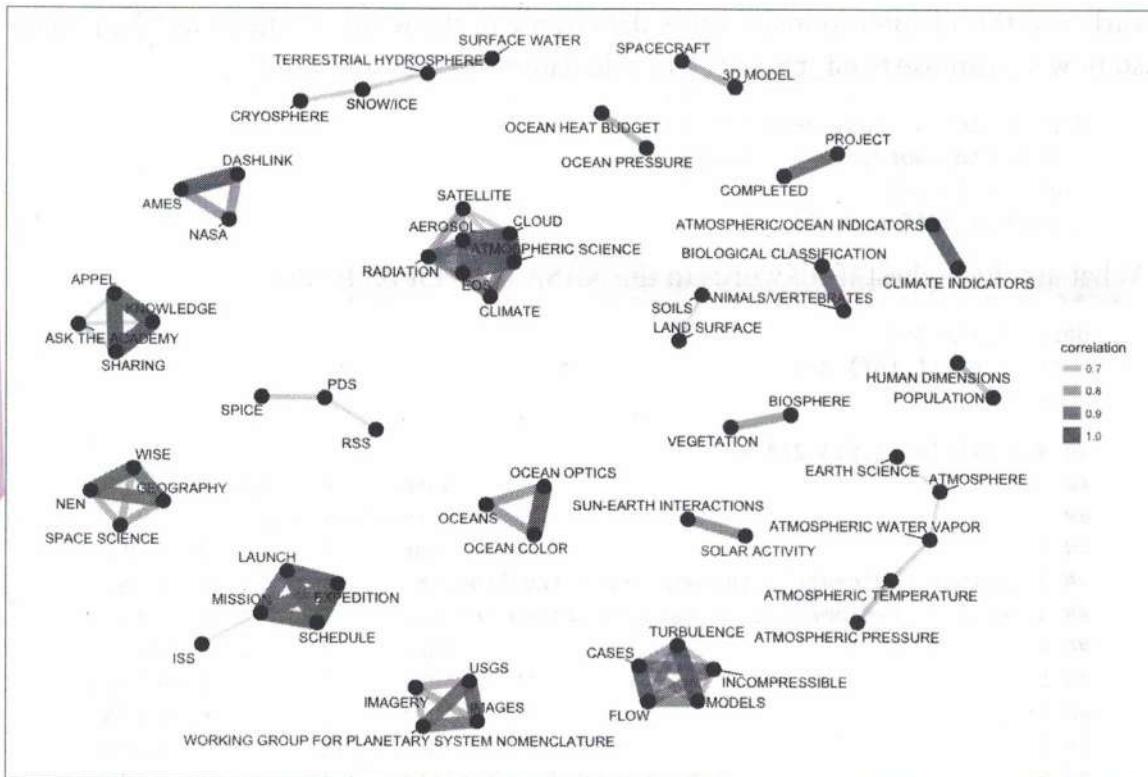


Figure 8-4. Correlation network in NASA dataset keywords

This network in Figure 8-4 appears much different than the co-occurrence network. The difference is that the co-occurrence network asks a question about which keyword pairs occur most often, and the correlation network asks a question about which keywords occur more often together than with other keywords. Notice here the high number of small clusters of keywords; the network structure can be extracted (for further analysis) from the `graph_from_data_frame()` function above.

## Calculating tf-idf for the Description Fields

The network graph in Figure 8-2 showed us that the description fields are dominated by a few common words like “data,” “global,” and “resolution”; this would be an excellent opportunity to use tf-idf as a statistic to find characteristic words for individual description fields. As discussed in Chapter 3, we can use tf-idf, the term frequency times inverse document frequency, to identify words that are especially important to a document within a collection of documents. Let’s apply that approach to the description fields of these NASA datasets.

### What Is tf-idf for the Description Field Words?

We will consider each description field a document, and the whole set of description fields the collection or corpus of documents. We have already used `unnest_tokens()`



earlier in this chapter to make a tidy data frame of the words in the description fields, so now we can use `bind_tf_idf()` to calculate tf-idf for each word.

```
desc_tf_idf <- nasa_desc %>%
  count(id, word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, id, n)
```

What are the highest tf-idf words in the NASA description fields?

```
desc_tf_idf %>%
  arrange(-tf_idf) %>%
  select(-id)

## # A tibble: 1,913,224 × 6
##   word     n      tf      idf
##   <chr> <int> <dbl>    <dbl>
## 1 rdr      1     1.0375052
## 2 palsar_radiometric_terrain_corrected_high_res 1     1.0375052
## 3 cpalsar_radiometric_terrain_corrected_low_res 1     1.0375052
## 4 lgrs     1     8.765615
## 5 lgrs     1     8.765615
## 6 lgrs     1     8.765615
## 7 mri      1     8.583293
## 8 template_proddescription 1     8.295611
## 9 template_proddescription 1     8.295611
## 10 template_proddescription 1     8.295611
## # ... with 1,913,214 more rows, and 1 more variables: tf_idf <dbl>
```

These are the most important words in the description fields as measured by tf-idf, meaning they are common but not too common.



Notice we have run into an issue here; both `n` and term frequency are equal to 1 for these terms, meaning that these were description fields that only had a single word in them. If a description field only contains one word, the tf-idf algorithm will think that is a very important word.

isn't it though? :-)

Depending on our analytic goals, it might be a good idea to throw out all description fields that have very few words.

## Connecting Description Fields to Keywords

We now know which words in the descriptions have high tf-idf, and we also have labels for these descriptions in the keywords. Let's do a full join of the keyword data frame and the data frame of description words with tf-idf, and then find the highest tf-idf words for a given keyword.

```
desc_tf_idf <- full_join(desc_tf_idf, nasa_keyword, by = "id")
```

Let's plot some of the most important words, as measured by tf-idf, for a few example keywords used on NASA datasets. First, let's use dplyr operations to filter for the keywords we want to examine and take just the top 15 words for each keyword. Then, let's plot those words in Figure 8-5.

```
desc_tf_idf %>%
  filter(!near(tf, 1)) %>%
  filter(keyword %in% c("SOLAR ACTIVITY", "CLOUDS",
                        "SEISMOLOGY", "ASTROPHYSICS",
                        "HUMAN HEALTH", "BUDGET")) %>%
  arrange(desc(tf_idf)) %>%
  group_by(keyword) %>%
  distinct(word, keyword, .keep_all = TRUE) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  ggplot(aes(word, tf_idf, fill = keyword)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~keyword, ncol = 3, scales = "free") +
  coord_flip() +
  labs(title = "Highest tf-idf words in NASA metadata description fields",
       caption = "NASA metadata from https://data.nasa.gov/data.json",
       x = NULL, y = "tf-idf")
```

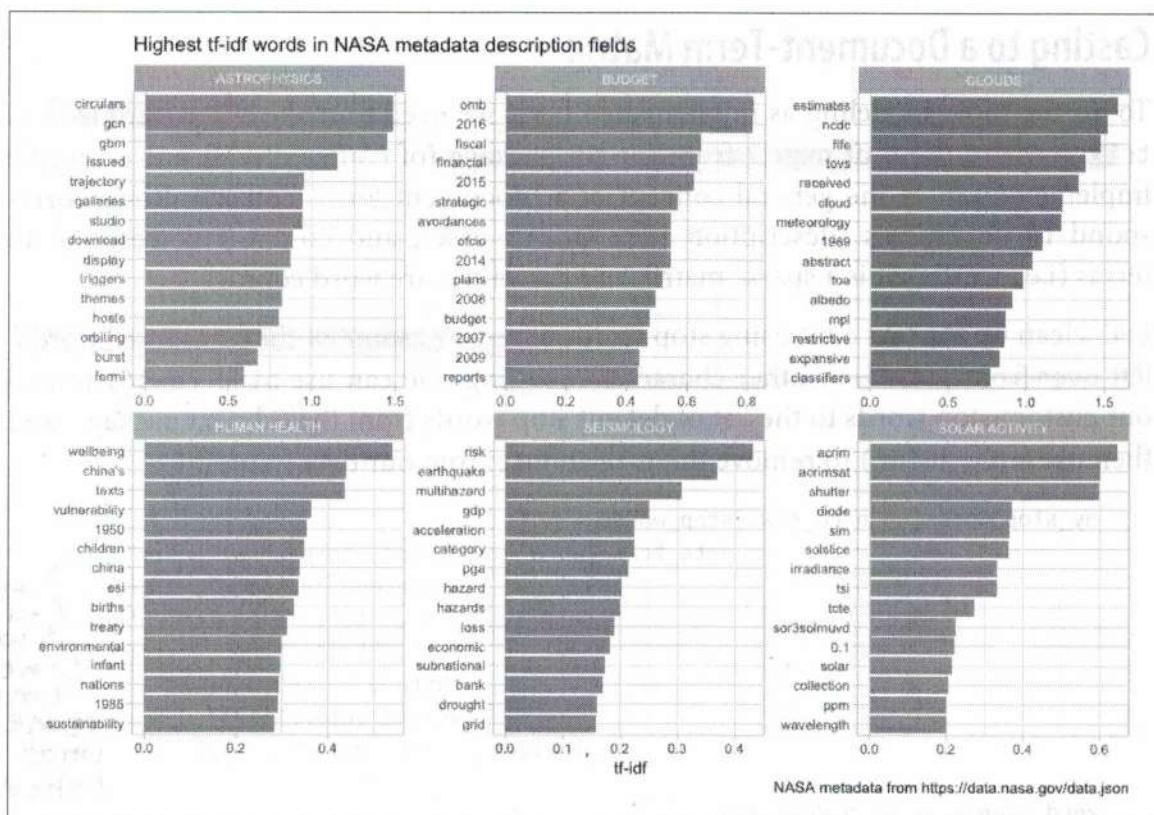


Figure 8-5. Distribution of tf-idf for words from datasets labeled with select keywords

Using tf-idf has allowed us to identify important description words for each of these keywords. Datasets labeled with the keyword “SEISMOLOGY” have words like “earthquake,” “risk,” and “hazard” in their description, while those labeled with “HUMAN HEALTH” have descriptions characterized by words like “wellbeing,” “vulnerability,” and “children.” Most of the combinations of letters that are not English words are certainly acronyms (like OMB for the Office of Management and Budget), and the examples of years and numbers are important for these topics. The tf-idf statistic has identified the kinds of words it is intended to—important words for individual documents within a collection of documents.

## Topic Modeling

Using tf-idf as a statistic has already given us insight into the content of NASA description fields, but let’s try an additional approach to the question of what the NASA descriptions fields are about. We can use topic modeling, as described in Chapter 6, to model each document (description field) as a mixture of topics, and each topic as a mixture of words. As in earlier chapters, we will use latent Dirichlet allocation (LDA) for our topic modeling; there are other possible approaches for topic modeling.

### Casting to a Document-Term Matrix

To do the topic modeling as implemented here, we need to make a `DocumentTermMatrix`, a special kind of matrix from the `tm` package (of course, this is just a specific implementation of the general concept of a “document-term matrix”). Rows correspond to documents (description texts in our case), and columns correspond to terms (i.e., words); it is a sparse matrix and the values are word counts.

Let’s clean up the text a bit using stop words to remove some of the nonsense “words” left over from HTML or other character encoding. We can use `bind_rows()` to add our custom stop words to the list of default stop words from the `tidytext` package, and then use `anti_join()` to remove them all at once from our data frame.

```
my_stop_words <- bind_rows(stop_words,
                           data_frame(word = c("nbsp", "amp", "gt", "lt",
                                               "timesnewromanpsmt", "font",
                                               "td", "li", "br", "tr", "quot",
                                               "st", "img", "src", "strong",
                                               "http", "file", "files",
                                               as.character(1:12)),
                           lexicon = rep("custom", 30)))
```

it'd be nice  
to know  
what these  
mean so next  
time I can  
remember  
what stop words  
I should include.

```
word_counts <- nasa_desc %>%
  anti_join(my_stop_words) %>%
  count(id, word, sort = TRUE) %>%
  ungroup()
```

```
word_counts

## # A tibble: 1,895,310 × 3
## # ... with 1,895,300 more rows
```

This is the information we need, the number of times each word is used in each document, to make a DocumentTermMatrix. We can `cast()` from our tidy text format to this nontidy format, as described in detail in Chapter 5.

```
desc_dtm <- word_counts %>%
  cast_dtm(id, word, n)

desc_dtm

## <<DocumentTermMatrix (documents: 32003, terms: 35901)>>
## Non-/sparse entries: 1895310/1147044393
## Sparsity           : 100%
## Maximal term length: 166
## Weighting          : term frequency (tf)
```

We see that this dataset contains documents (each of them a NASA description field) and terms (words). Notice that this example document-term matrix is (very close to) 100% sparse, meaning that almost all of the entries in this matrix are zero. Each non-zero entry corresponds to a certain word appearing in a certain document.

# Ready for Topic Modeling

Now let's use the `topicmodels` package to create an LDA model. How many topics will we tell the algorithm to make? This is a question much like in  $k$ -means clustering; we don't really know ahead of time. We tried the following modeling procedure using 8, 16, 24, 32, and 64 topics; we found that at 24 topics, documents are still getting sorted into topics cleanly, but going much beyond that caused the distributions of  $\gamma$ , the probability that each document belongs in each topic, to look worrisome. We will show more details on this later.

```

library(topicmodels)

# be aware that running this model is time intensive
desc_lda <- LDA(desc_dtm, k = 24, control = list(seed = 1234))
desc_lda

## A LDA_VEM topic model with 24 topics.

```

This is a stochastic algorithm that could have different results depending on where the algorithm starts, so we need to specify a seed for reproducibility as shown here.

## Interpreting the Topic Model

Now that we have built the model, let's `tidy()` the results of the model, i.e., construct a tidy data frame that summarizes the results of the model. The `tidytext` package includes a tidying method for LDA models from the `topicmodels` package.

```

tidy_lda <- tidy(desc_lda)

tidy_lda

## # A tibble: 861,624 × 3
##   topic term      beta
##   <int> <chr>    <dbl>
## 1     1 suit 1.003981e-121
## 2     2 suit 2.630614e-145
## 3     3 suit 1.916240e-79
## 4     4 suit 6.715725e-45
## 5     5 suit 1.738334e-85
## 6     6 suit 7.692116e-84
## 7     7 suit 3.283851e-04
## 8     8 suit 3.738586e-20
## 9     9 suit 4.846953e-15
## 10    10 suit 4.765471e-10
## # ... with 861,614 more rows

```

The column  $\beta$  tells us the probability of that term being generated from that topic for that document. It is the probability of that term (word) belonging to that topic. Notice that some of the values for  $\beta$  are very, very low, and some are not so low.

What is each topic about? Let's examine the top 10 terms for each topic.

```

top_terms <- tidy_lda %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms

## # A tibble: 240 × 3
##   topic term      beta
##   <int> <chr>    <dbl>
## 1     1 suit 1.003981e-121
## 2     1 suit 2.630614e-145
## 3     1 suit 1.916240e-79
## 4     1 suit 6.715725e-45
## 5     1 suit 1.738334e-85
## 6     1 suit 7.692116e-84
## 7     1 suit 3.283851e-04
## 8     1 suit 3.738586e-20
## 9     1 suit 4.846953e-15
## 10    1 suit 4.765471e-10
## # ... with 239 more rows

```

```

## 1      1      data 0.04488960
## 2      1      soil 0.03676198
## 3      1      moisture 0.02954555
## 4      1      amsr 0.02437751
## 5      1      sst 0.01684001
## 6      1 validation 0.01322457
## 7      1 temperature 0.01317075
## 8      1 surface 0.01290046
## 9      1 accuracy 0.01225131
## 10     1      set 0.01155372
## # ... with 230 more rows

```

It is not very easy to interpret what the topics are about from a data frame like this, so let's look at this information visually in Figures 8-6 and 8-7. super helpful!

Looking @ #s isn't great...

```

top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  group_by(topic, term) %>%
  arrange(desc(beta)) %>%
  ungroup() %>%
  mutate(term = factor(paste(term, topic, sep = "__"),
                      levels = rev(paste(term, topic, sep = "__")))) %>%
  ggplot(aes(term, beta, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  scale_x_discrete(labels = function(x) gsub("__.+$", "", x)) +
  labs(title = "Top 10 terms in each LDA topic",
       x = NULL, y = expression(beta)) +
  facet_wrap(~ topic, ncol = 3, scales = "free")

```

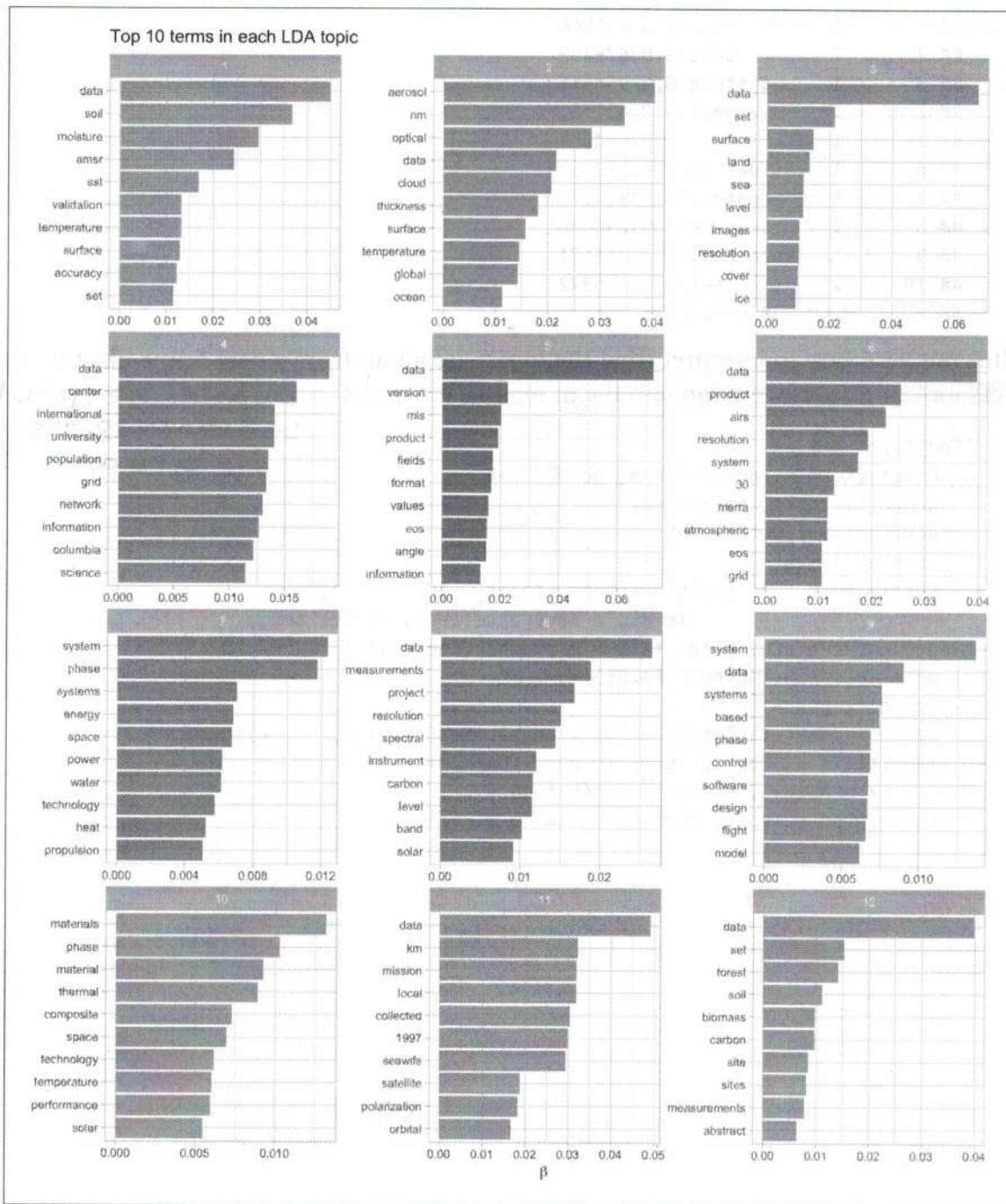


Figure 8-6. Top terms in topic modeling of NASA metadata description field texts

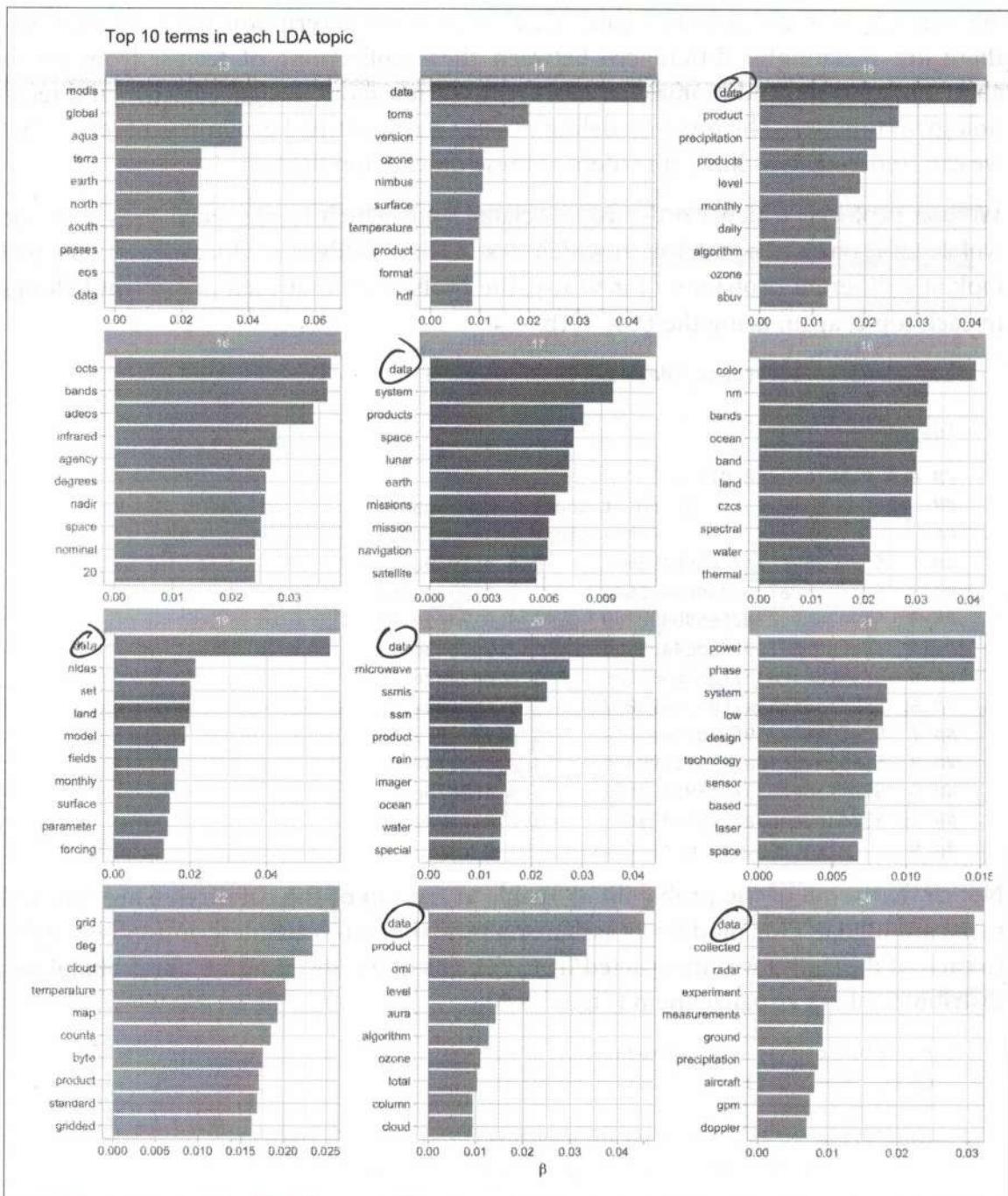


Figure 8-7. Top terms in topic modeling of NASA metadata description field texts

We can see what a dominant word “data” is in these description texts. In addition, there are meaningful differences between these collections of terms, from terms about soil, forests, and biomass in topic 12 to terms about design, systems, and technology in topic 21. The topic modeling process has identified groupings of terms that we can understand as human readers of these description fields.

We just explored which words are associated with which topics. Next, let's examine which topics are associated with which description fields (i.e., documents). We will look at a different probability for this,  $\gamma$ , the probability that each document belongs in each topic, again using the `tidy` verb.

```
lda_gamma <- tidy(desc_lda, matrix = "gamma")  
  
lda_gamma  
  
## # A tibble: 768,072 × 3  
##   document topic      gamma  
##   <chr>     <int>    <dbl>  
## 1 55942a8ec63a7fe59b4986ef     1 6.453820e-06  
## 2 56cf5b00a759fdadc44e564a     1 1.158393e-05  
## 3 55942a89c63a7fe59b4982d9     1 4.917441e-02  
## 4 56cf5b00a759fdadc44e55cd     1 2.249043e-05  
## 5 55942a89c63a7fe59b4982c6     1 6.609442e-05  
## 6 55942a86c63a7fe59b498077     1 5.666520e-05  
## 7 56cf5b00a759fdadc44e56f8     1 4.752082e-05  
## 8 55942a8bc63a7fe59b4984b5     1 4.308534e-05  
## 9 55942a6ec63a7fe59b496bf7     1 4.408626e-05  
## 10 55942a8ec63a7fe59b4986f6    1 2.878188e-05  
## # ... with 768,062 more rows
```

Notice that some of the probabilities visible at the top of the data frame are low and some are higher. Our model has assigned a probability to each description belonging to each of the topics we constructed from the sets of words. How are the probabilities distributed? Let's visualize them (Figure 8-8).

```
ggplot(lda_gamma, aes(gamma)) +  
  geom_histogram() +  
  scale_y_log10() +  
  labs(title = "Distribution of probabilities for all topics",  
       y = "Number of documents", x = expression(gamma))
```

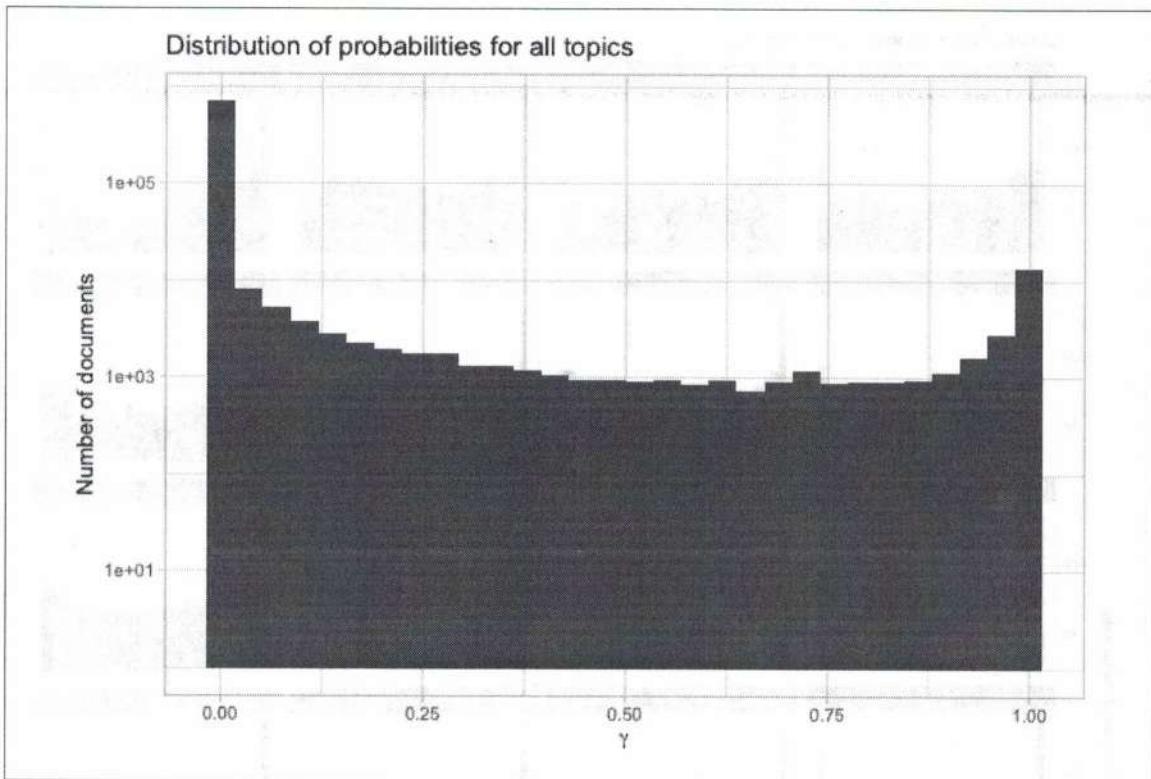


Figure 8-8. Probability distribution in topic modeling of NASA metadata description field texts

First notice that the y-axis is plotted on a log scale; otherwise it is difficult to make out any detail in the plot. Next, notice that  $\gamma$  runs from 0 to 1; remember that this is the probability that a given document belongs in a given topic. There are many values near zero, which means there are many documents that do not belong in each topic. Also, there are many values near  $\gamma = 1$ ; these are the documents that do belong in those topics. This distribution shows that documents are being well discriminated as belonging to a topic or not. We can also look at how the probabilities are distributed within each topic, as shown in Figure 8-9.

```
ggplot(lda_gamma, aes(gamma, fill = as.factor(topic))) +
  geom_histogram(show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 4) +
  scale_y_log10() +
  labs(title = "Distribution of probability for each topic",
       y = "Number of documents", x = expression(gamma))
```

→ ie topics  
are  
sometimes  
super  
specific  
  
related only  
to one  
document  
in the set of  
documents

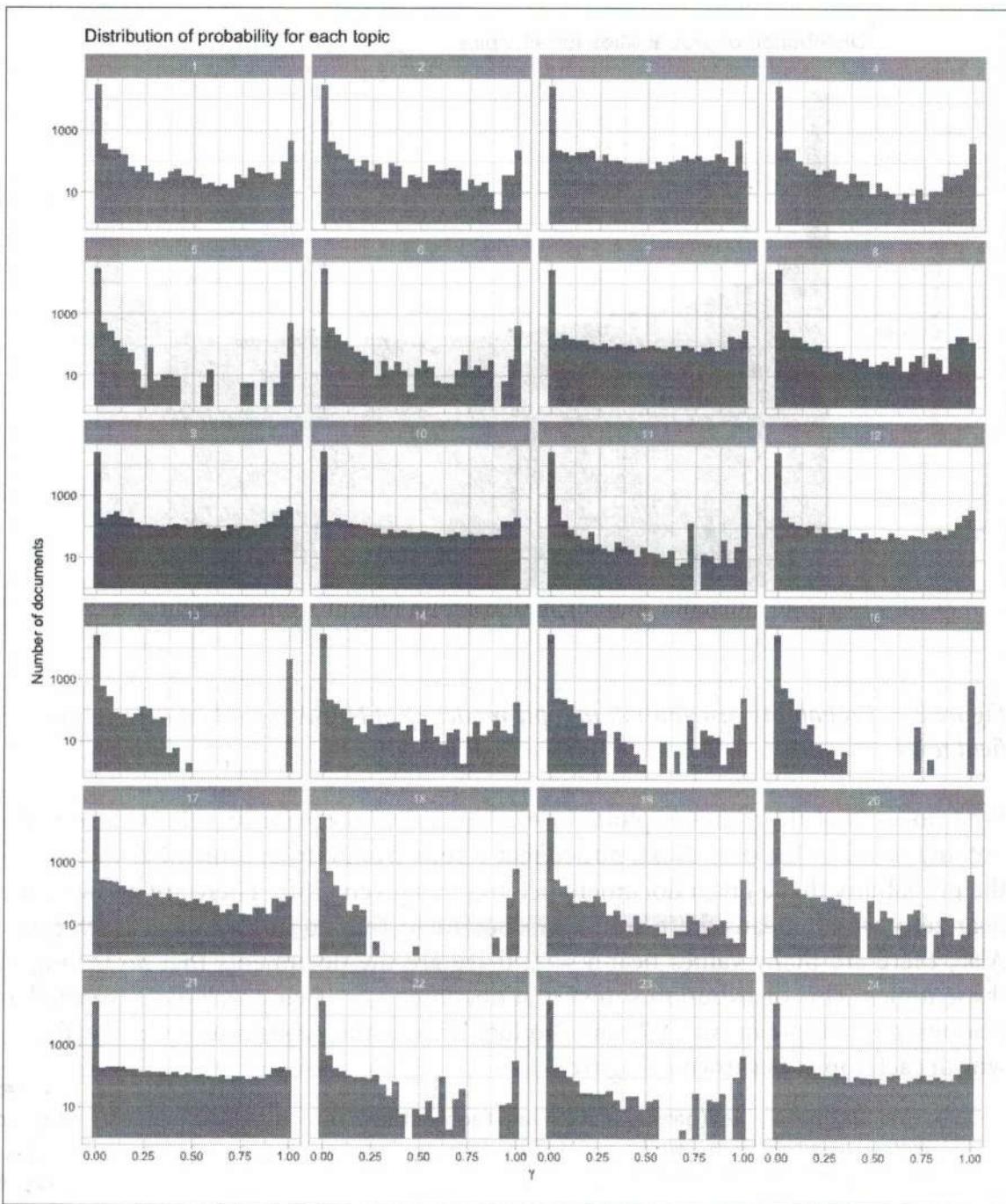


Figure 8-9. Probability distribution for each topic in topic modeling of NASA metadata description field texts

Let's look specifically at topic 18 in Figure 8-9, a topic that had documents cleanly sorted in and out of it. There are many documents with  $y$  close to 1; these are the documents that *do* belong to topic 18 according to the model. There are also many documents with  $y$  close to 0; *these are the documents that do not belong to topic 18.*

Each document appears in each panel in this plot, and its  $\gamma$  for that topic tells us that document's probability of belonging in that topic.

This plot displays the type of information we used to choose the number of topics for our topic modeling procedure. When we tried options higher than 24 (such as 32 or 64), the distributions for  $\gamma$  started to look very flat toward  $\gamma = 1$ ; documents were **not** getting sorted into topics very well.

## Connecting Topic Modeling with Keywords

Let's connect these topic models with the keywords and see what relationships we can find. We can `full_join()` this to the human-tagged keywords and discover which keywords are associated with which topic.

```
lda_gamma <- full_join(lda_gamma, nasa_keyword, by = c("document" = "id"))

lda_gamma

## # A tibble: 3,037,671 × 4
##   document topic      gamma      keyword
##   <chr>    <int>     <dbl>     <chr>
## 1 55942a8ec63a7fe59b4986ef    1 6.453820e-06 JOHNSON SPACE CENTER
## 2 55942a8ec63a7fe59b4986ef    1 6.453820e-06 PROJECT
## 3 55942a8ec63a7fe59b4986ef    1 6.453820e-06 COMPLETED
## 4 56cf5b00a759fdadc44e564a    1 1.158393e-05 DASHLINK
## 5 56cf5b00a759fdadc44e564a    1 1.158393e-05 AMES
## 6 56cf5b00a759fdadc44e564a    1 1.158393e-05 NASA
## 7 55942a89c63a7fe59b4982d9    1 4.917441e-02 GODDARD SPACE FLIGHT CENTER
## 8 55942a89c63a7fe59b4982d9    1 4.917441e-02 PROJECT
## 9 55942a89c63a7fe59b4982d9    1 4.917441e-02 COMPLETED
## 10 56cf5b00a759fdadc44e55cd  1 2.249043e-05 DASHLINK
## # ... with 3,037,661 more rows
```

Now we can use `filter()` to keep only the document-topic entries that have probabilities ( $\gamma$ ) greater than some cutoff value; let's use 0.9.  
so 90% likely or higher

```
top_keywords <- lda_gamma %>%
  filter(gamma > 0.9) %>%
  count(topic, keyword, sort = TRUE)

top_keywords

## Source: local data frame [1,022 × 3]
## Groups: topic [24]
##
##   topic      keyword     n
##   <int>     <chr> <int>
## 1    13      OCEAN COLOR 4480
## 2    13      OCEAN OPTICS 4480
## 3    13      OCEANS 4480
## 4    11      OCEAN COLOR 1216
## 5    11      OCEAN OPTICS 1216
```

```
## 6    11      OCEANS  1216
## 7     9      PROJECT  926
## 8    12 EARTH SCIENCE  909
## 9     9      COMPLETED 834
## 10   16 OCEAN COLOR  768
## # ... with 1,012 more rows
```

What are the top keywords for each topic (Figure 8-10)?

```
top_keywords %>%
  group_by(topic) %>%
  top_n(5, n) %>%
  group_by(topic, keyword) %>%
  arrange(desc(n)) %>%
  ungroup() %>%
  mutate(keyword = factor(paste(keyword, topic, sep = "___"),
                         levels = rev(paste(keyword, topic, sep = "___")))) %>%
  ggplot(aes(keyword, n, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  labs(title = "Top keywords for each LDA topic",
       x = NULL, y = "Number of documents") +
  coord_flip() +
  scale_x_discrete(labels = function(x) gsub("___+$", "", x)) +
  facet_wrap(~ topic, ncol = 3, scales = "free")
```

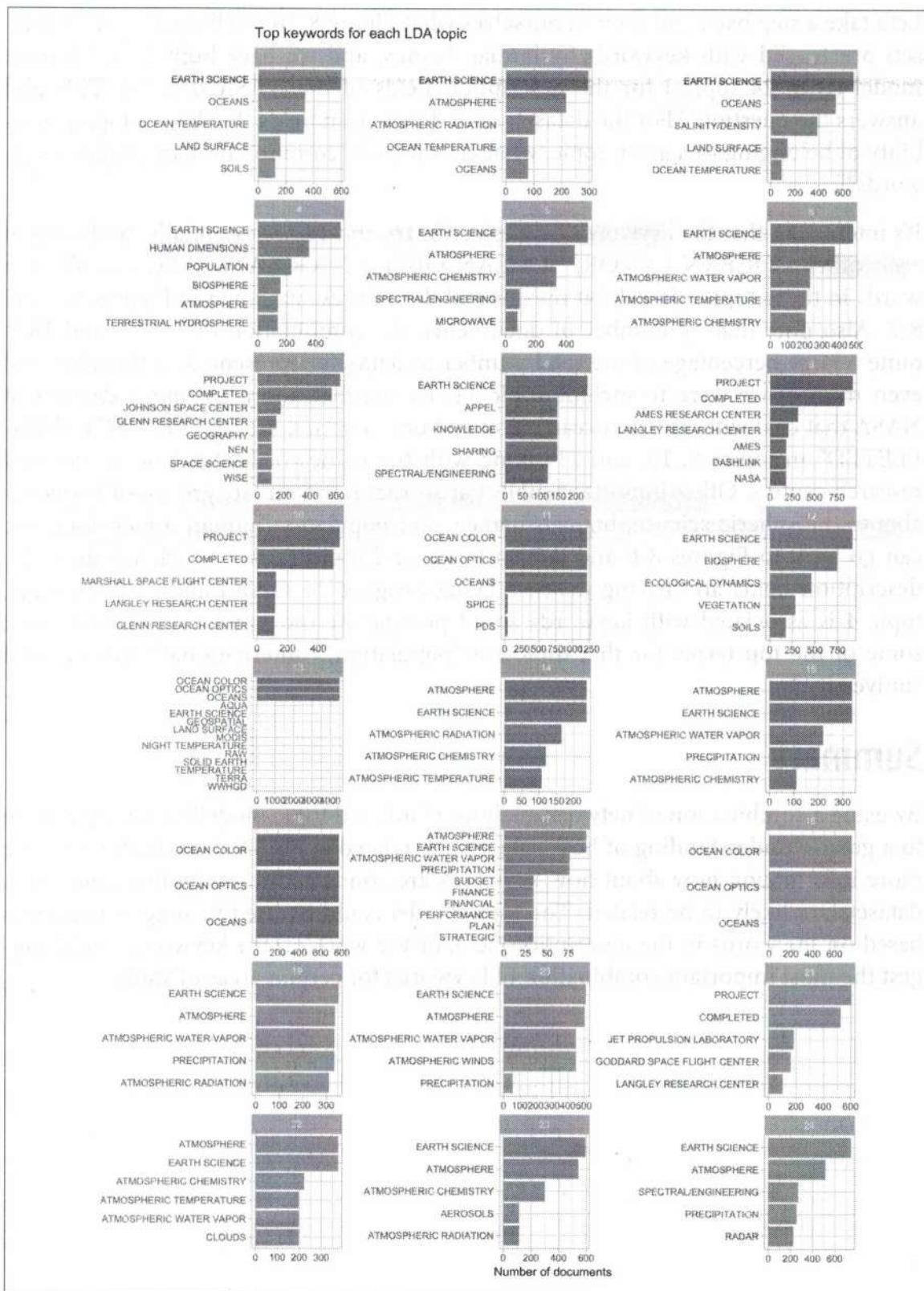


Figure 8-10. Top keywords in topic modeling of NASA metadata description field texts

Let's take a step back and remind ourselves what Figure 8-10 is telling us. NASA datasets are tagged with keywords by human beings, and we have built an LDA topic model (with 24 topics) for the description fields of the NASA datasets. This plot answers the question, "For the datasets with description fields that have a high probability of belonging to a given topic, what are the most common human-assigned keywords?"

It's interesting that the **keywords for topics 13, 16, and 18 are essentially duplicates of each other** ("OCEAN COLOR," "OCEAN OPTICS," "OCEANS"), because the top words in those topics do exhibit meaningful differences, as shown in Figures 8-6 and 8-7. Also note that by number of documents, the combination of 13, 16, and 18 is quite a large percentage of the total number of datasets represented in this plot, and even more if we were to include topic 11. By number, there are *many* datasets at NASA that deal with oceans, ocean color, and ocean optics. We see "PROJECT COMPLETED" in topics 9, 10, and 21, along with the names of NASA laboratories and research centers. Other important subject areas that stand out are groups of keywords about atmospheric science, budget/finance, and population/human dimensions. We can go back to Figures 8-6 and 8-7 on terms and topics to see which words in the description fields are driving datasets being assigned to these topics. For example, topic 4 is associated with keywords about population and human dimensions, and some of the top terms for that topic are "population," "international," "center," and "university."

## Summary

By using a combination of network analysis, tf-idf, and topic modeling, we have come to a greater understanding of how datasets are related at NASA. Specifically, we have more information now about how keywords are connected to each other and which datasets are likely to be related. The topic model could be used to suggest keywords based on the words in the description field, or the work on the keywords could suggest the most important combination of keywords for certain areas of study.

## CHAPTER 9

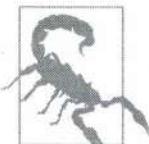
# Case Study: Analyzing Usenet Text

In our final chapter, we'll use what we've learned in this book to perform a start-to-finish analysis of a set of 20,000 messages sent to 20 Usenet bulletin boards in 1993. The Usenet bulletin boards in this dataset include newsgroups for topics like politics, religion, cars, sports, and cryptography, and offer a rich set of text written by many users. This data set is publicly available at <http://qwone.com/~jason/20Newsgroups/> (the *20news-bydate.tar.gz* file) and has become popular for exercises in text analysis and machine learning.

## Preprocessing

Step 1) Download folder, save in your location  
Step 2) Read in with your custom path

We'll start by reading in all the messages from the *20news-bydate* folder, which are organized in subfolders with one file for each message. We can read in files like these with a combination of `read_lines()`, `map()`, and `unnest()`.



Note that this step may take several minutes to read all the documents.

```
library(dplyr)
library(tidyr)
library(purrr)
library(readr)

training_folder <- "data/20news-bydate/20news-bydate-train/"

# Define a function to read all files from a folder into a data frame
read_folder <- function(infolder) {
  data_frame(file = dir(infolder, full.names = TRUE)) %>%
```

```

    mutate(text = map(file, read_lines)) %>%
    transmute(id = basename(file), text) %>%
    unnest(text)
}

# Use unnest() and map() to apply read_folder to each subfolder
raw_text <- data_frame(folder = dir(training_folder, full.names = TRUE)) %>%
  unnest(map(folder, read_folder)) %>%
  transmute(newsgroup = basename(folder), id, text)

raw_text

# A tibble: 511,655 x 3
  newsgroup     id
  <chr> <chr>
1 alt.atheism 49960
2 alt.atheism 49960
3 alt.atheism 49960
4 alt.atheism 49960
5 alt.atheism 49960
6 alt.atheism 49960
7 alt.atheism 49960
8 alt.atheism 49960
9 alt.atheism 49960
10 alt.atheism 49960
# ... with 511,645 more rows, and 1 more variables: text <chr>

```

Notice the `newsgroup` column, which describes which of the 20 newsgroups each message comes from, and the `id` column, which identifies a unique message within that `newsgroup`. What newsgroups are included, and how many messages were posted in each (Figure 9-1)?

```

library(ggplot2)

raw_text %>%
  group_by(newsgroup) %>%
  summarize(messages = n_distinct(id)) %>%
  ggplot(aes(newsgroup, messages)) +
  geom_col() +
  coord_flip()

```

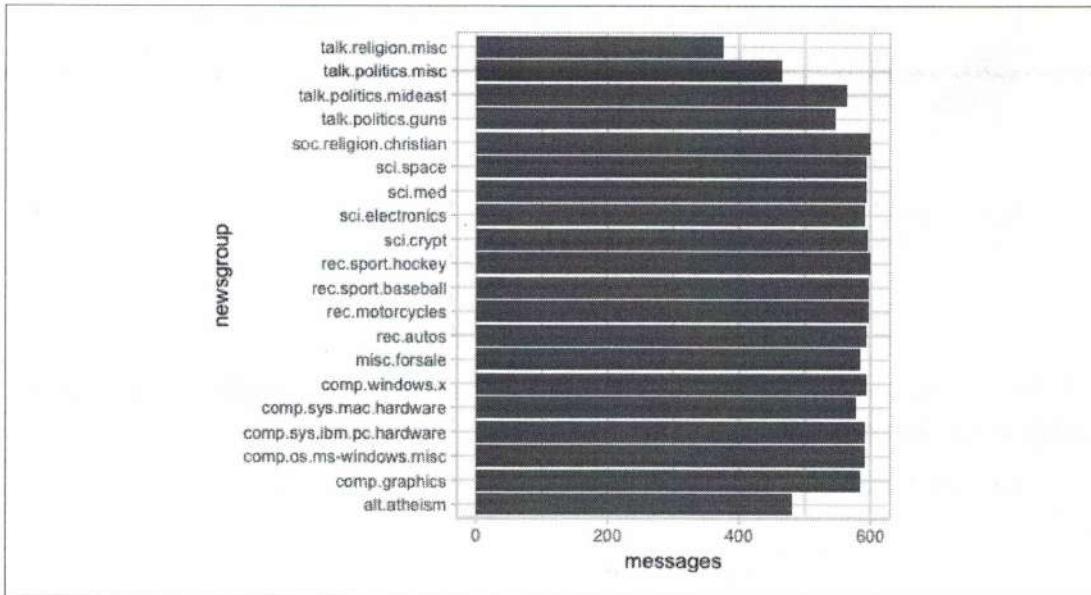


Figure 9-1. Number of messages from each newsgroup

We can see that Usenet newsgroup names are named hierarchically, starting with a main topic such as “talk,” “sci,” or “rec,” followed by further specifications.

## Preprocessing Text

Most of the datasets we've examined in this book were preprocessed, meaning we didn't have to remove, for example, copyright notices from the Jane Austen novels. ~~Each message has some structure and extra text that we don't want to~~

~~analyze. For example, every message has a header containing fields such as From: or To: before the actual text that contains the message body. We can remove these signatures which occur after a line like~~ ↗ by asking R to detect them

This kind of preprocessing can be done within the dplyr package, using a combination of cumsum() (cumulative sum) and str\_detect() from stringr.

```
library(stringr)

# must occur after the first occurrence of an empty line,
# and before the first occurrence of a line starting with --
cleaned_text <- raw_text %>%
  group_by(newsgroup, id) %>%
  filter(cumsum(text == "") > 0,
         cumsum(str_detect(text, "^--")) == 0) %>%
  ungroup()
```

Many lines also have nested text representing quotes from other users, typically starting with a line like “so-and-so writes...” These can be removed with a few regular expressions.

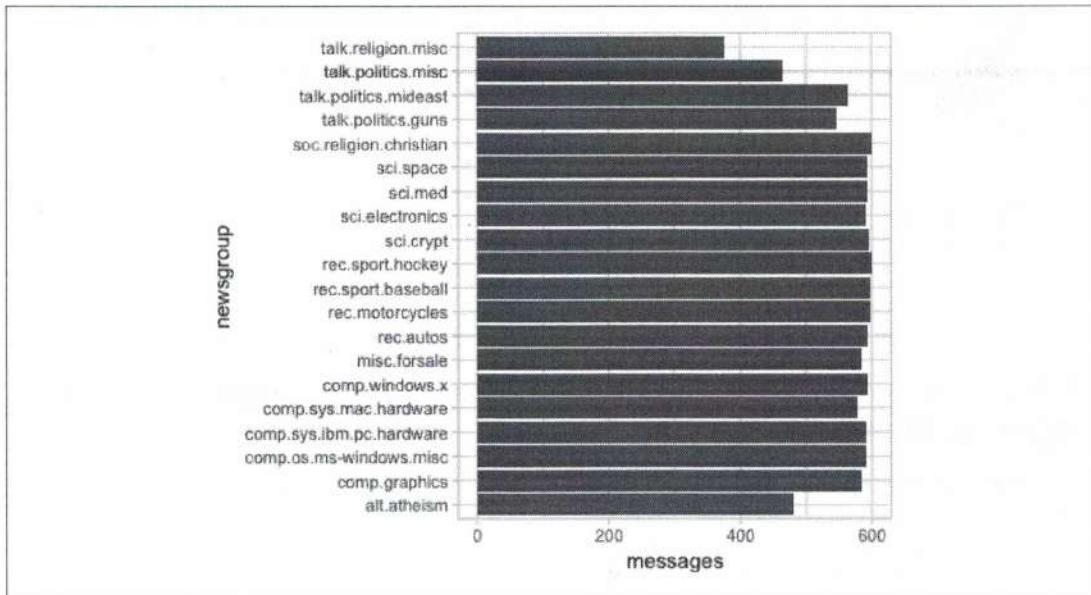


Figure 9-1. Number of messages from each newsgroup

We can see that Usenet newsgroup names are named hierarchically, starting with a main topic such as “talk,” “sci,” or “rec,” followed by further specifications.

## Preprocessing Text

Most of the datasets we've examined in this book were preprocessed, meaning we didn't have to remove, for example, copyright notices from the Jane Austen novels. Here, however, each message has some structure and extra text that we don't want to include in our analysis. For example, every message has a header containing fields such as “from:” or “in\_reply\_to:” that describe the message. Some also have automated email signatures, which occur after a line like ...

so we can remove these  
by asking R to detect them

This kind of preprocessing can be done within the dplyr package, using a combination of `cumsum()` (cumulative sum) and `str_detect()` from stringr.

```
library(stringr)

# must occur after the first occurrence of an empty line,
# and before the first occurrence of a line starting with --
cleaned_text <- raw_text %>%
  group_by(newsgroup, id) %>%
  filter(cumsum(text == "") > 0,
         cumsum(str_detect(text, "^--")) == 0) %>%
  ungroup()
```

Many lines also have nested text representing quotes from other users, typically starting with a line like “so-and-so writes...” These can be removed with a few regular expressions.



We also choose to manually remove two messages, 9704 and 9985, that contain a large amount of nontext content.

```
cleaned_text <- cleaned_text %>%
  filter(str_detect(text, "^[^>]+[A-Za-z\\d]") | text == "",  
        !str_detect(text, "writes(:|\\.\\.\\.\\.$"),  
        !str_detect(text, "^In article <"),  
        !id %in% c(9704, 9985))
```

At this point, we're ready to use `unnest_tokens()` to split the dataset into tokens, while removing stop words.

```
library(tidytext)  
  
usenet_words <- cleaned_text %>%
  unnest_tokens(word, text) %>% ✓  
  filter(str_detect(word, "[a-z']$"), ✓  
        !word %in% stop_words$word) ✓
```

Every raw text dataset will require different steps for data cleaning, which will often involve some trial and error, and exploration of unusual cases in the dataset. It's important to notice that this cleaning can be achieved using tidy tools such as `dplyr` and `tidyverse`.

## Words in Newsgroups

Now that we've removed the headers, signatures, and formatting, we can start exploring common words. For starters, we could find the most common words in the entire dataset or within particular newsgroups.

```
usenet_words %>%
  count(word, sort = TRUE) ✓  
  
## # A tibble: 68,137 × 2  
##       word     n  
##       <chr> <int>  
## 1   people  3655  
## 2      time  2705  
## 3      god  1626  
## 4    system  1595  
## 5   program  1103  
## 6      bit  1097  
## 7 information  1094  
## 8    windows  1088  
## 9 government  1084  
## 10     space  1072  
## # ... with 68,127 more rows
```

```

words_by_newsgroup <- usenet_words %>%
  count(newsgroup, word, sort = TRUE) %>%
  ungroup()

words_by_newsgroup

## # A tibble: 173,913 × 3
##   newsgroup     word     n
##   <chr>       <chr> <int>
## 1 soc.religion.christian god    917
## 2 sci.space      space   840
## 3 talk.politics.mideast people 728
## 4 sci.crypt      key    704
## 5 comp.os.ms-windows.misc windows 625
## 6 talk.politics.mideast armenian 582
## 7 sci.crypt      db     549
## 8 talk.politics.mideast turkish 514
## 9 rec.autos       car    509
## 10 talk.politics.mideast armenians 509
## # ... with 173,903 more rows

```

## Finding tf-idf Within Newsgroups

We'd expect the newsgroups to differ in terms of topic and content, and therefore for the frequency of words to differ between them. Let's try quantifying this using the tf-idf metric (Chapter 3).

```

tf_idf <- words_by_newsgroup %>%
  bind_tf_idf(word, newsgroup, n) %>%
  arrange(desc(tf_idf))

tf_idf

# A tibble: 173,913 × 6
#   newsgroup     word     n      tf      idf
#   <chr>       <chr> <int>   <dbl>   <dbl>
1 comp.sys.ibm.pc.hardware scsi    483 0.01761681 1.20397
2 talk.politics.mideast armenian 582 0.00804890 2.30259
3 rec.motorcycles        bike    324 0.01389842 1.20397
4 talk.politics.mideast armenians 509 0.00703933 2.30259
5 sci.crypt              encryption 410 0.00816099 1.89712
6 rec.sport.hockey        nhl    157 0.00439665 2.99573
7 talk.politics.misc     stephanopoulos 158 0.00416228 2.99573
8 rec.motorcycles        bikes   97 0.00416095 2.99573
9 rec.sport.hockey        hockey  270 0.00756112 1.60944
10 comp.windows.x         oname   136 0.00353550 2.99573
# ... with 173,903 more rows, and 1 more variables: tf_idf <dbl>

```

We can examine the top tf-idf for a few selected groups to extract words specific to those topics. For example, we could look at all the sci. boards, visualized in Figure 9-2.

```

tf_idf %>%
  filter(str_detect(newsgroup, "sci\\\\."))
  group_by(newsgroup) %>%
  top_n(12, tf_idf) %>%
  ungroup() %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(word, tf_idf, fill = newsgroup)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ newsgroup, scales = "free") +
  ylab("tf-idf") +
  coord_flip()

```

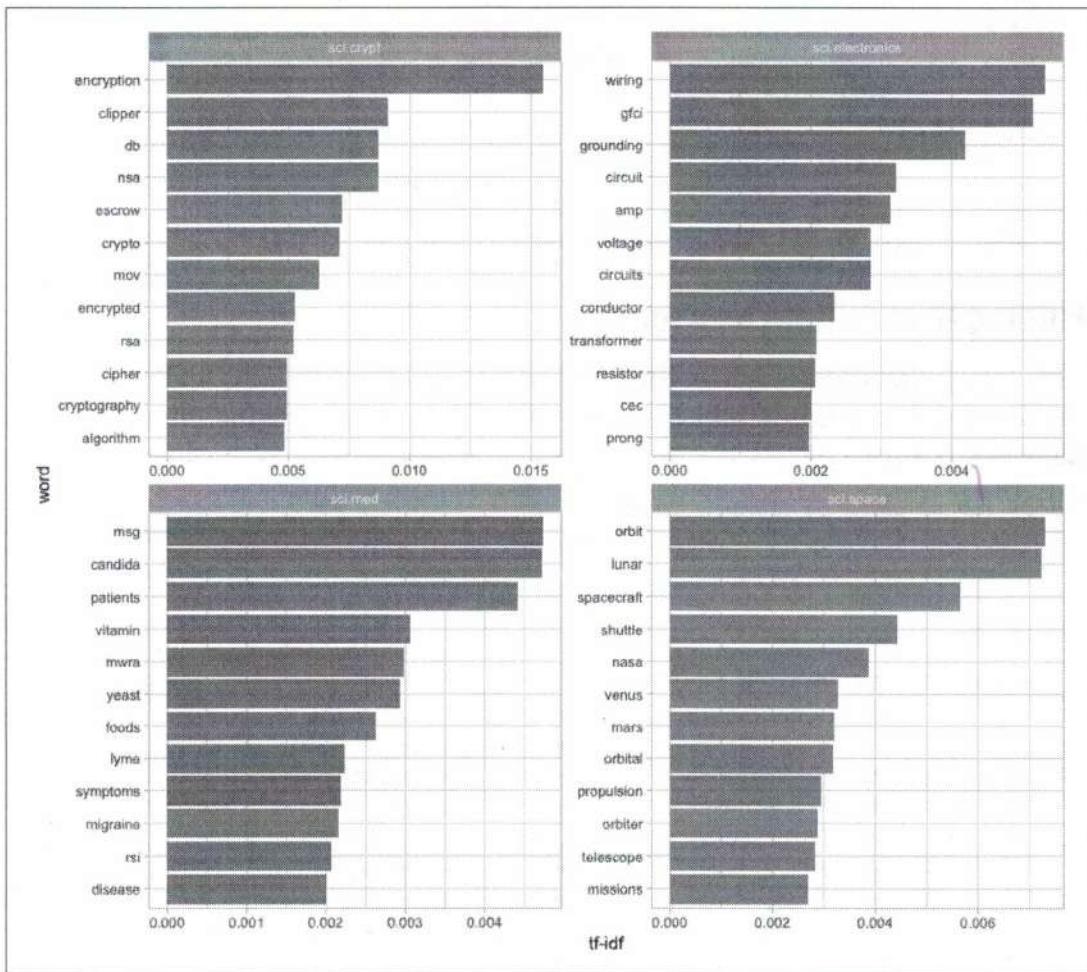


Figure 9-2. The 12 terms with the highest tf-idf within each of the science-related newsgroups

We see lots of characteristic words specific to a particular newsgroup, such as “wiring” and “circuit” on the `sci.electronics` topic, and “orbit” and “lunar” for the space newsgroup. You could use this same code to explore other newsgroups yourself.

What newsgroups tend to be similar to each other in text content? We could discover this by finding the pairwise correlation of word frequencies within each newsgroup, using the `pairwise_cor()` function from the `widyr` package (see “Examining Pairwise Correlation” on page 63).

```
library(widyr)

newsgroup_cors <- words_by_newsgroup %>%
  pairwise_cor(newsgroup, word, n, sort = TRUE)

newsgroup_cors

## # A tibble: 380 × 3
##       item1           item2 correlation
##       <chr>          <chr>      <dbl>
## 1 talk.religion.misc soc.religion.christian 0.8347275
## 2 soc.religion.christian talk.religion.misc 0.8347275
## 3 alt.atheism         talk.religion.misc 0.7793079
## 4 talk.religion.misc   alt.atheism        0.7793079
## 5 alt.atheism         soc.religion.christian 0.7510723
## 6 soc.religion.christian alt.atheism        0.7510723
## 7 comp.sys.mac.hardware comp.sys.ibm.pc.hardware 0.6799043
## 8 comp.sys.ibm.pc.hardware comp.sys.mac.hardware 0.6799043
## 9 rec.sport.baseball    rec.sport.hockey   0.5770378
## 10 rec.sport.hockey     rec.sport.baseball  0.5770378
## # ... with 370 more rows
```

We could then filter for stronger correlations among newsgroups and visualize them in a network (Figure 9-3).

```
library(ggraph)
library(igraph)
set.seed(2017)

newsgroup_cors %>%
  filter(correlation > .4) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = correlation, width = correlation)) +
  geom_node_point(size = 6, color = "lightblue") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```

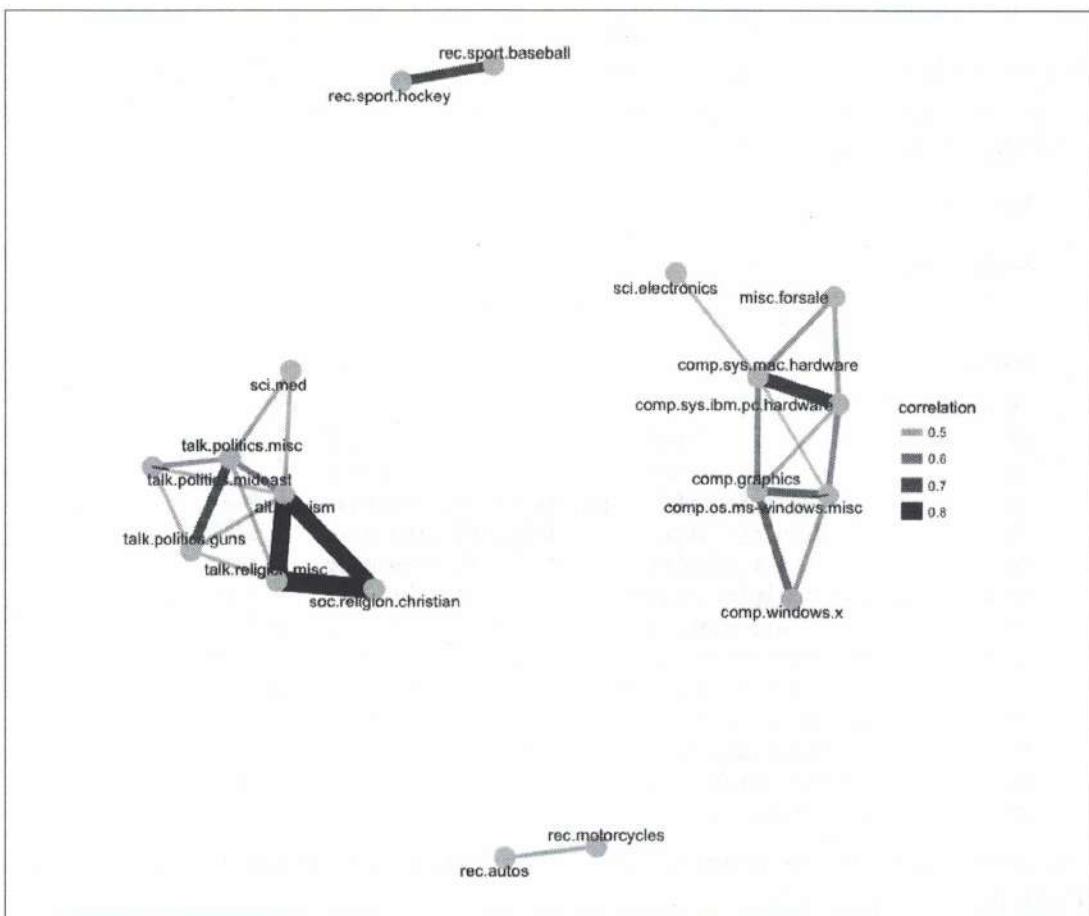


Figure 9-3. A network of Usenet groups based on the correlation of word counts between them, including only connections with a correlation greater than 0.4

It looks like there are four main clusters of newsgroups: computers/electronics, politics/religion, motor vehicles, and sports. This certainly makes sense in terms of what words and topics we'd expect these newsgroups to have in common.

## Topic Modeling

In Chapter 6, we used the latent Dirichlet allocation (LDA) algorithm to divide a set of chapters into the books they originally came from. Could LDA do the same to sort out Usenet messages that come from different newsgroups?

Let's try dividing up messages from the four science-related newsgroups. We first process these into a document-term matrix with `cast_dtm()` ("Casting Tidy Text Data into a Matrix" on page 77), then fit the model with the `LDA()` function from the `topicmodels` package.

```
# include only words that occur at least 50 times
word_sci_newsgroups <- usenet_words %>%
```

```

filter(str_detect(newsgroup, "^sci")) %>%
group_by(word) %>%
mutate(word_total = n()) %>%
ungroup() %>%
filter(word_total > 50)

# convert into a document-term matrix
# with document names such as sci.crypt_14147
sci_dtm <- word_sci_newsgroups %>%
unite(document, newsgroup, id) %>%
count(document, word) %>%
cast_dtm(document, word, n)

library(topicmodels)
sci_lda <- LDA(sci_dtm, k = 4, control = list(seed = 2016))

```

**What four topics did this model extract, and do they match the four newsgroups?**  
This approach will look familiar from Chapter 6: we visualize each topic based on the most frequent terms within it (Figure 9-4).

```

sci_lda %>%
tidy() %>%
group_by(topic) %>%
top_n(8, beta) %>%
ungroup() %>%
mutate(term = reorder(term, beta)) %>%
ggplot(aes(term, beta, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
facet_wrap(~ topic, scales = "free_y") +
coord_flip()

```

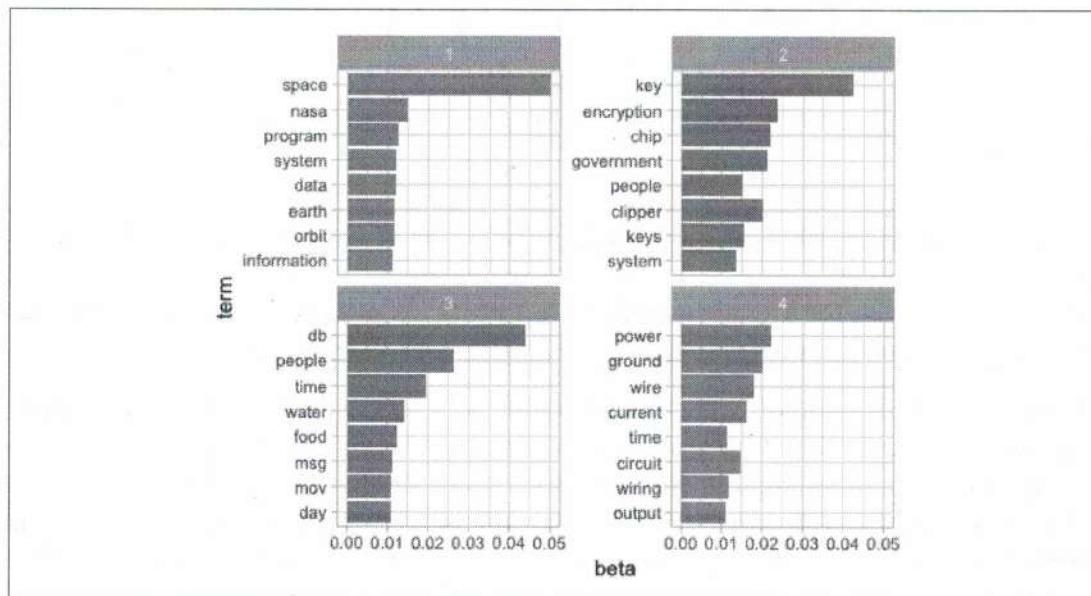


Figure 9-4. The top eight words from each topic fit by LDA on the science-related newsgroups

From the top words, we can start to suspect which topics may capture which newsgroups. Topic 1 certainly represents the sci.space newsgroup (thus the most common word being “space”), and topic 2 is likely drawn from cryptography, with terms such as “key” and “encryption.” Just as we did in “Document-Topic Probabilities” on page 95, we can confirm this by seeing how documents from each newsgroup have higher “gamma” for each topic (Figure 9-5).

```
sci_lda %>%
  tidy(matrix = "gamma") %>%
  separate(document, c("newsgroup", "id"), sep = "_") %>%
  mutate(newsgroup = reorder(newsgroup, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ newsgroup) +
  labs(x = "Topic",
       y = "# of messages where this was the highest % topic")
```

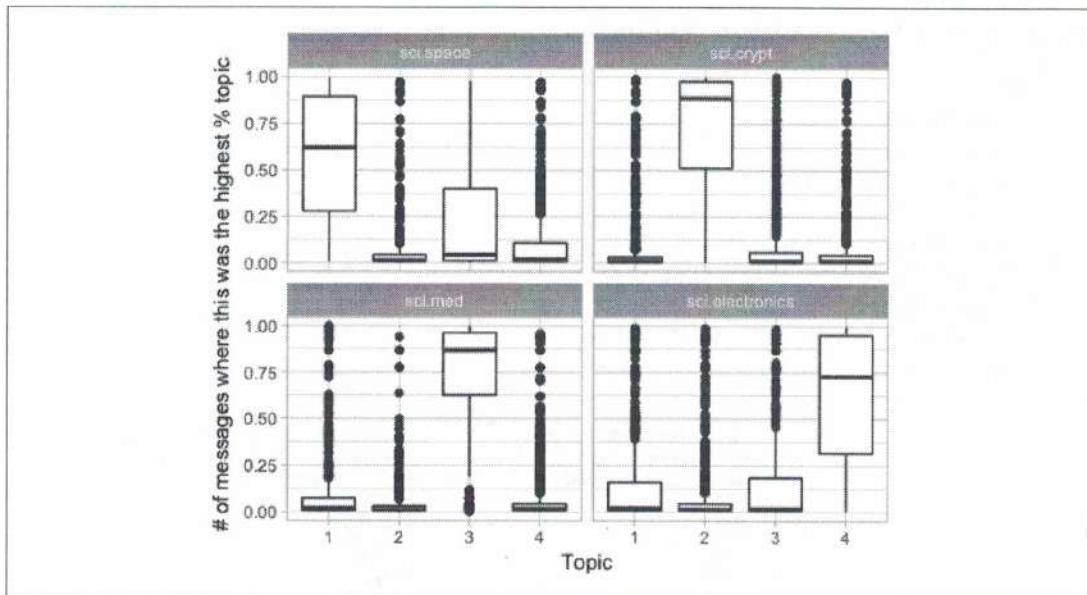


Figure 9-5. Distribution of gamma for each topic within each Usenet newsgroup

Much as we saw in the literature analysis, topic modeling was able to discover the distinct topics present in the text without needing to consult the labels.

Notice that the division of Usenet messages isn’t as clean as the division of book chapters, with a substantial number of messages from each newsgroup getting high values of “gamma” for other topics. This isn’t surprising since many of the messages are short and could overlap in terms of common words (for example, discussions of space travel could include many of the same words as discussions of electronics). This is a realistic example of how LDA might divide documents into rough topics while still allowing a degree of overlap.

## Sentiment Analysis

We can use the sentiment analysis techniques we explored in Chapter 2 to examine how often positive and negative words occur in these Usenet posts. Which newsgroups are the most positive or negative overall?

In this example we'll use the AFINN sentiment lexicon, which provides numeric positivity scores for each word, and visualize it with a bar plot (Figure 9-6).

```
newsgroup_sentiments <- words_by_newsgroup %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(newsgroup) %>%
  summarize(score = sum(score * n) / sum(n))

newsgroup_sentiments %>%
  mutate(newsgroup = reorder(newsgroup, score)) %>%
  ggplot(aes(newsgroup, score, fill = score > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  ylab("Average sentiment score")
```

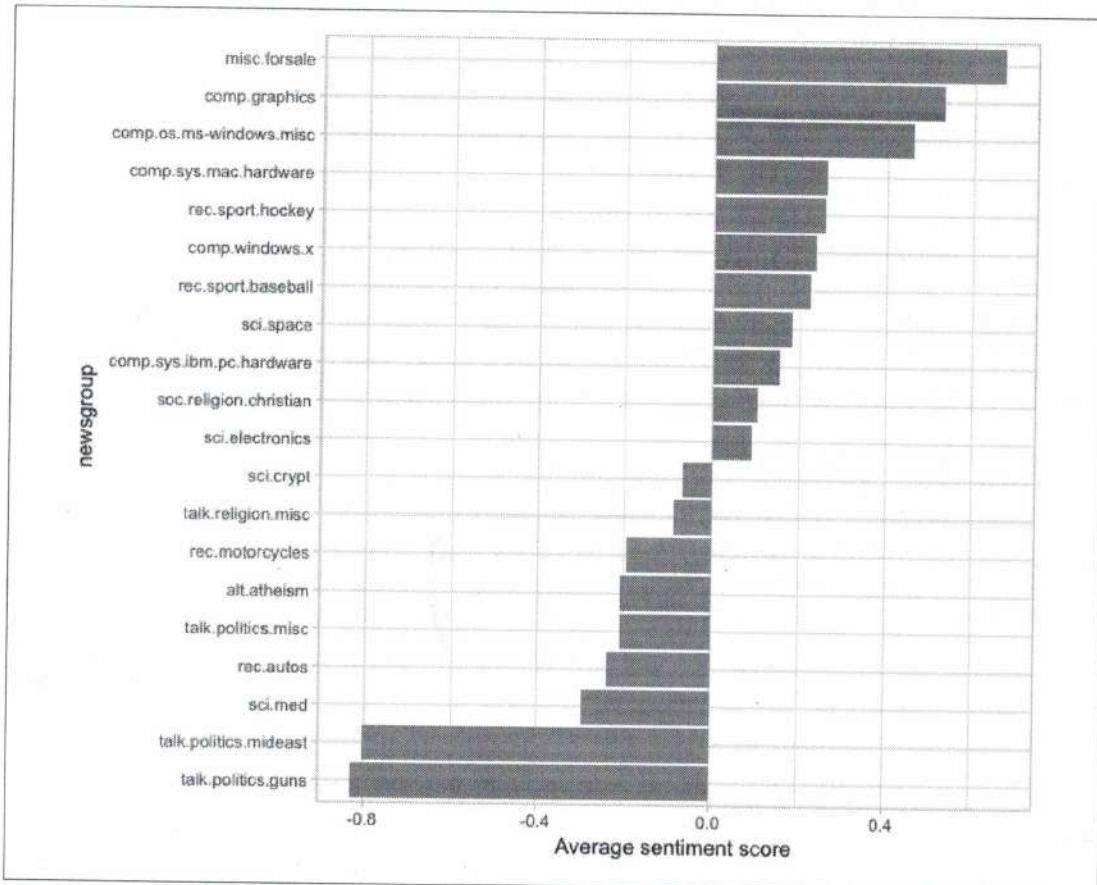


Figure 9-6. Average AFINN score for posts within each newsgroup

According to this analysis, the misc.forsale newsgroup is the most positive. This makes sense, since it likely includes many positive adjectives about the products that users want to sell!

## Sentiment Analysis by Word

It's worth looking deeper to understand *why* some newsgroups end up more positive or negative than others. For that, we can examine the total positive and negative contributions of each word. *this is useful for checking biases*

```
contributions <- usenet_words %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(word) %>%
  summarize(occurrences = n(),
            contribution = sum(score))

contributions

## # A tibble: 1,909 × 3
##       word occurrences contribution
##   <chr>     <int>        <int>
## 1 abandon      13        -26
## 2 abandoned    19        -38
## 3 abandons      3         -6
## 4 abduction     2         -4
## 5 abhor        4        -12
## 6 abhorred     1         -3
## 7 abhorrent    2         -6
## 8 abilities     16         32
## 9 ability       177        354
## 10 aboard       8          8
## # ... with 1,899 more rows
```

Which words have the most effect on sentiment scores overall (Figure 9-7)?

```
contributions %>%
  top_n(25, abscontribution) %>%
  mutate(word = reorder(word, contribution)) %>%
  ggplot(aes(word, contribution, fill = contribution > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip()
```

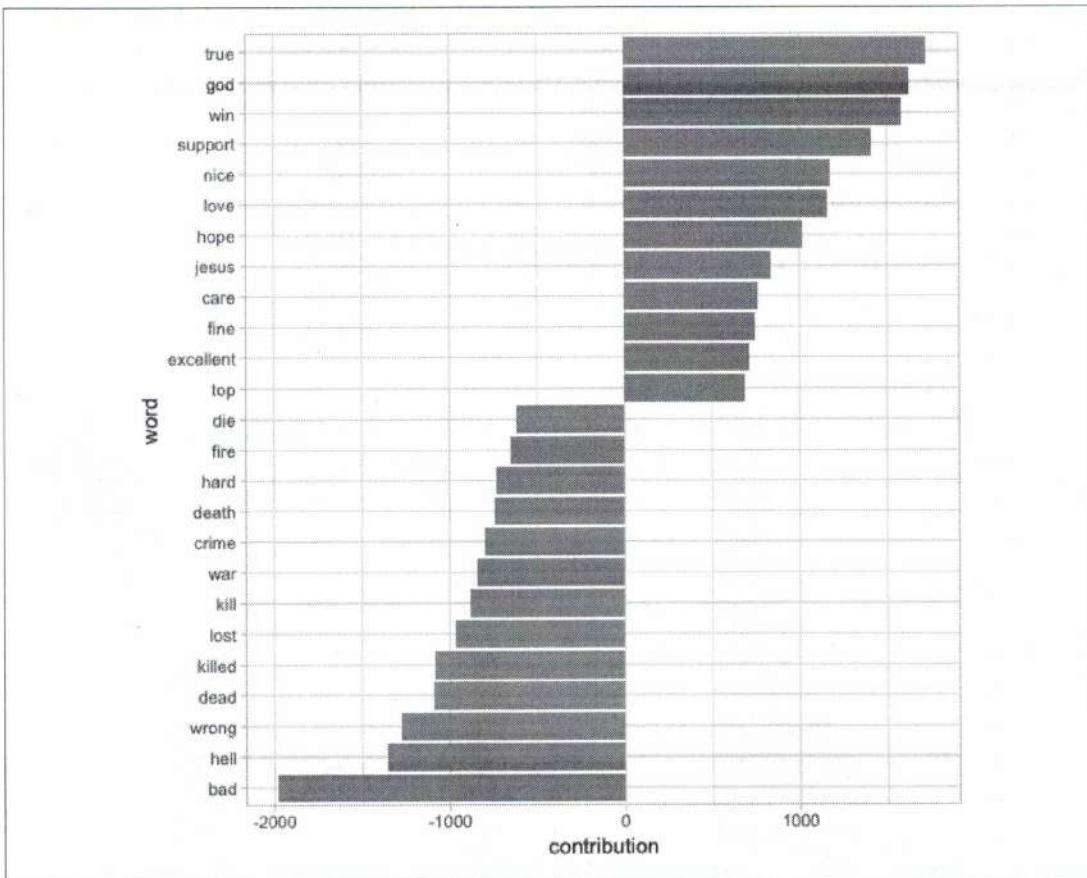


Figure 9-7. Words with the greatest contributions to positive/negative sentiment scores in the Usenet text

These words look generally reasonable as indicators of each message's sentiment, but we can spot possible problems with the approach. "True" could just as easily be a part of "not true" or a similar negative expression, and the words "God" and "Jesus" are apparently very common on Usenet but could easily be used in many contexts, positive or negative.

We may also care about which words contribute the most *within each newsgroup*, so that we can see which newsgroups might be incorrectly estimated. We can calculate each word's contribution to each newsgroup's sentiment score, and visualize the strongest contributors from a selection of the groups (Figure 9-8).

```
top_sentiment_words <- words_by_newsgroup %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  mutate(contribution = score * n / sum(n))

top_sentiment_words
## # A tibble: 13,063 × 5
##       newsgroup     word     n   score contribution
```

```

##           <chr>  <chr> <int> <int>      <dbl>
## 1 soc.religion.christian god    917    1  0.014418012
## 2 soc.religion.christian jesus   440    1  0.006918130
## 3 talk.politics.guns   gun    425   -1 -0.006682285
## 4 talk.religion.misc   god    296    1  0.004654015
## 5 alt.atheism          god    268    1  0.004213770
## 6 soc.religion.christian faith  257    1  0.004040817
## 7 talk.religion.misc   jesus   256    1  0.004025094
## 8 talk.politics.mideast killed 202   -3 -0.009528152
## 9 talk.politics.mideast war    187   -2 -0.005880411
## 10 soc.religion.christian true   179    2  0.005628842
## # ... with 13,053 more rows

```

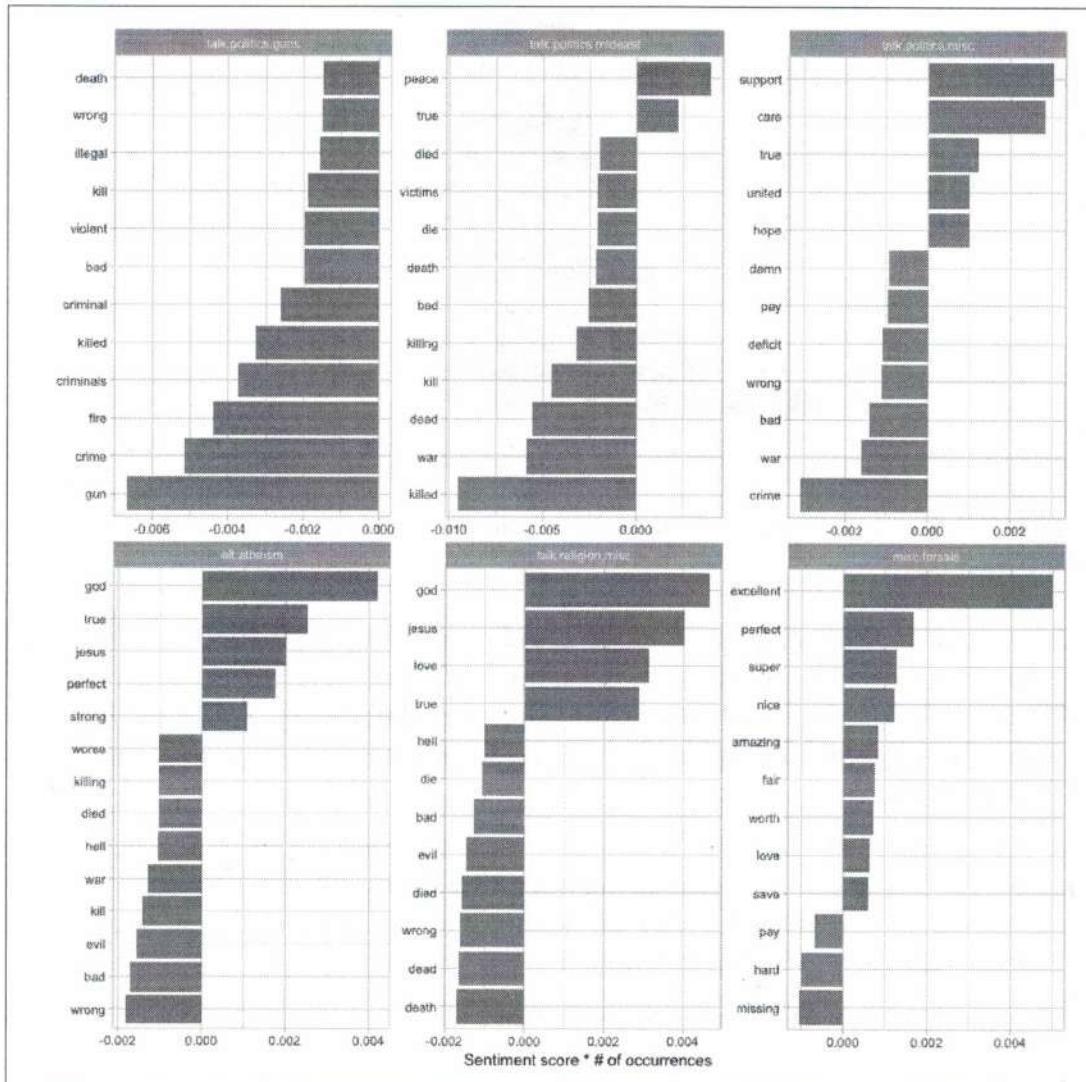


Figure 9-8. The 12 words that contributed the most to sentiment scores within each of 6 newsgroups

This confirms our hypothesis about the misc.forsale newsgroup: most of the sentiment is driven by positive adjectives such as “excellent” and “perfect.” We can also see how much sentiment is confounded with topic. An atheism newsgroup is likely to discuss “god” in detail even in a negative context, and we can see that it makes the newsgroup look more positive. Similarly, the negative contribution of the word “gun” to the talk.politics.guns group will occur even when the members are discussing guns positively.

This helps remind us that sentiment analysis can be confounded by topic, and that we should always examine the influential words before interpreting the analysis too deeply.

## Sentiment Analysis by Message

We can also try finding the most positive and negative individual messages by grouping and summarizing by `id` rather than `newsgroup`.

```
sentiment_messages <- usenet_words %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(newsgroup, id) %>%
  summarize(sentiment = mean(score),
            words = n()) %>%
  ungroup() %>%
  filter(words >= 5)
```



As a simple measure to reduce the role of randomness, we filtered out messages that had fewer than five words that contributed to sentiment.

What were the most positive messages?

```
sentiment_messages %>%
  arrange(desc(sentiment))

## # A tibble: 3,554 × 4
##       newsgroup     id sentiment words
##       <chr>      <chr>     <dbl> <int>
## 1 rec.sport.hockey 53560  3.888889   18
## 2 rec.sport.hockey 53602  3.833333   30
## 3 rec.sport.hockey 53822  3.833333    6
## 4 rec.sport.hockey 53645  3.230769   13
## 5 rec.autos        102768  3.200000    5
## 6 misc.forsale     75965  3.000000    5
## 7 misc.forsale     76037  3.000000    5
## 8 rec.sport.baseball 104458  3.000000   11
## 9 rec.sport.hockey 53571  3.000000    5
## 10 comp.os.ms-windows.misc 9620  2.857143    7
## # ... with 3,544 more rows
```

Let's check this by looking at the most positive message in the whole dataset. To assist in this, we could write a short function for printing a specified message.

```
print_message <- function(group, message_id) {  
  result <- cleaned_text %>%  
    filter(newsgroup == group, id == message_id, text != "")  
  
  cat(result$text, sep = "\n")  
}  
  
print_message("rec.sport.hockey", 53560)  
  
## Everybody. Please send me your predictions for the Stanley Cup Playoffs!  
## I want to see who people think will win!!!!!!  
## Please Send them in this format, or something comparable:  
## 1. Winner of Buffalo-Boston  
## 2. Winner of Montreal-Quebec  
## 3. Winner of Pittsburgh-New York  
## 4. Winner of New Jersey-Washington  
## 5. Winner of Chicago-(Minnesota/St.Louis)  
## 6. Winner of Toronto-Detroit  
## 7. Winner of Vancouver-Winnipeg  
## 8. Winner of Calgary-Los Angeles  
## 9. Winner of Adams Division (1-2 above)  
## 10. Winner of Patrick Division (3-4 above)  
## 11. Winner of Norris Division (5-6 above)  
## 12. Winner of Smythe Division (7-8 above)  
## 13. Winner of Wales Conference (9-10 above)  
## 14. Winner of Campbell Conference (11-12 above)  
## 15. Winner of Stanley Cup (13-14 above)  
## I will summarize the predictions, and see who is the biggest  
## INTERNET GURU PREDICTING GUY/GAL.  
## Send entries to Richard Madison  
## rrmadiso@napier.uwaterloo.ca  
## PS: I will send my entries to one of you folks so you know when I say  
## I won, that I won!!!!!
```

It looks like this message was chosen because it uses the word "winner" many times. How about the most negative message? Turns out it's also from the hockey site, but has a very different attitude.

```
sentiment_messages %>%  
  arrange(sentiment)  
  
## # A tibble: 3,554 × 4  
##       newsgroup     id sentiment words  
##   <chr>     <chr>     <dbl> <int>  
## 1 rec.sport.hockey 53907 -3.000000  6  
## 2 sci.electronics 53899 -3.000000  5  
## 3 talk.politics.mideast 75918 -3.000000  7  
## 4 rec.autos 101627 -2.833333  6  
## 5 comp.graphics 37948 -2.800000  5  
## 6 comp.windows.x 67204 -2.700000 10
```

```

## 7      talk.politics.guns 53362 -2.666667    6
## 8          alt.atheism 51309 -2.600000    5
## 9  comp.sys.mac.hardware 51513 -2.600000    5
## 10         rec.autos 102883 -2.600000    5
## # ... with 3,544 more rows
print_message("rec.sport.hockey", 53907)

## Losers like us? You are the fucking moron who has never heard of the Western
## Business School, or the University of Western Ontario for that matter. Why
## don't you pull your head out of your asshole and smell something other than
## shit for once so you can look on a map to see where UWO is! Back to hockey,
## the North Stars should be moved because for the past few years they have
## just been SHIT. A real team like Toronto would never be moved!!!
## Andrew--

```

Well, we can confidently say that the sentiment analysis worked!

## N-gram Analysis

In Chapter 4, we considered the effect of words such as “not” and “no” on sentiment analysis of Jane Austen novels, such as considering whether a phrase like “don’t like” led to passages incorrectly being labeled as positive. The Usenet dataset is a much larger corpus of more modern text, so we may be interested in how sentiment analysis may be ~~reversed~~ in this text.

We’ll start by finding and counting all the bigrams in the Usenet posts.

```

usenet_bigrams <- cleaned_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

usenet_bigram_counts <- usenet_bigrams %>%
  count(newsgroup, bigram, sort = TRUE) %>%
  ungroup() %>%
  separate(bigram, c("word1", "word2"), sep = " ")

```

We could then define a list of six words that we suspect are used in negation, such as “no,” “not,” and “without,” and visualize the sentiment-associated words that most often follow them (Figure 9-9). This shows the words that most often contribute in the “wrong” direction.

```

negate_words <- c("not", "without", "no", "can't", "don't", "won't")

usenet_bigram_counts %>%
  filter(word1 %in% negate_words) %>%
  count(word1, word2, wt = n, sort = TRUE) %>%
  inner_join(get_sentiments("afinn"), by = c(word2 = "word")) %>%
  mutate(contribution = score * nn) %>%
  group_by(word1) %>%
  top_n(10, abs(contribution)) %>%
  ungroup() %>%
  mutate(word2 = reorder(paste(word2, word1, sep = " "), contribution)) %>%
  ggplot(aes(word2, contribution, fill = contribution > 0)) +

```

```

geom_col(show.legend = FALSE) +
facet_wrap(~ word1, scales = "free", nrow = 3) +
scale_x_discrete(labels = function(x) gsub("_.+$", "", x)) +
xlab("Words preceded by a negation") +
ylab("Sentiment score * # of occurrences") +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
coord_flip()

```

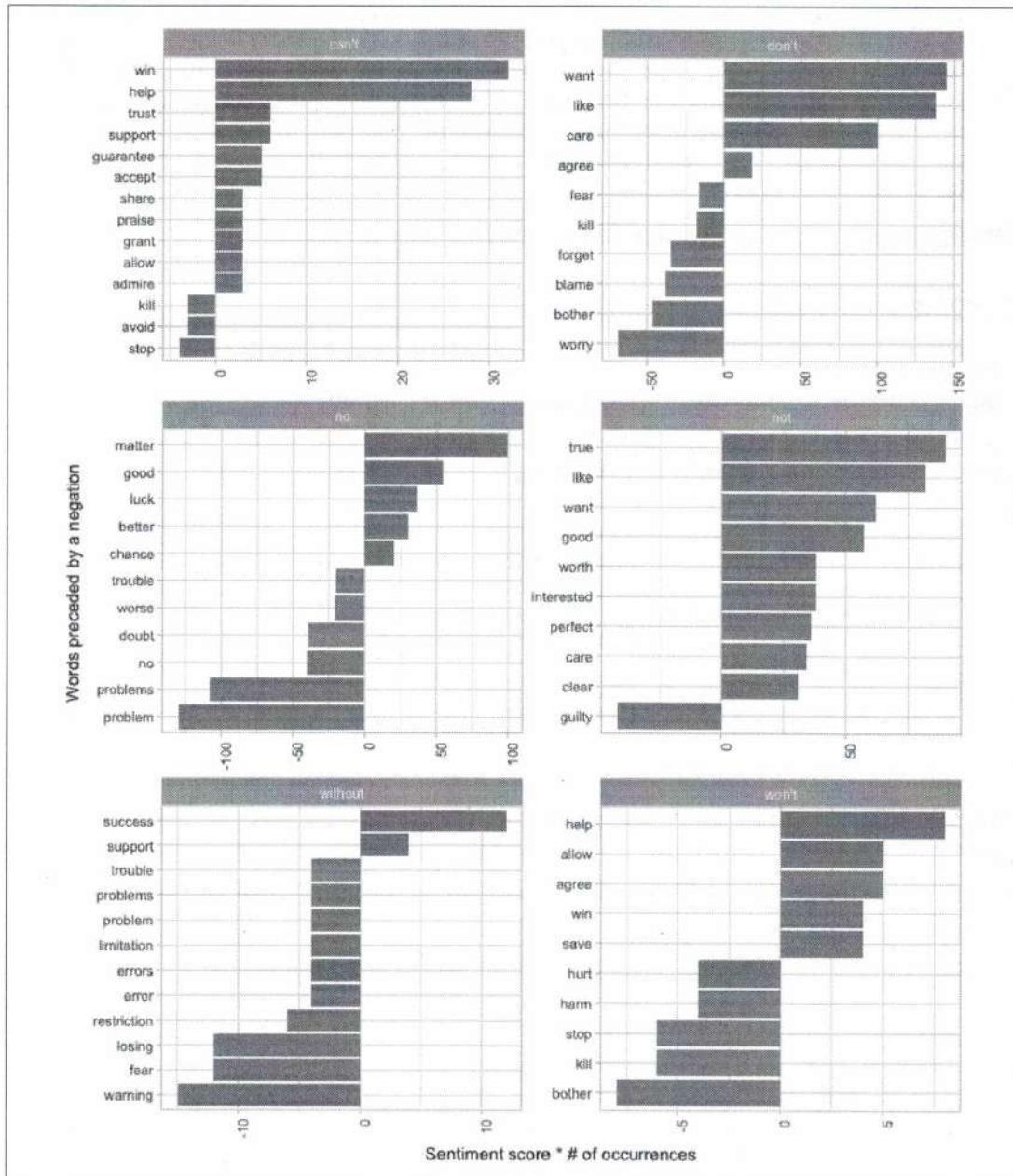


Figure 9-9. Words that contribute the most to sentiment when they follow a “negating” word

It looks like the largest sources of misidentifying a word as positive come from “don’t want/like/care,” and the largest source of incorrectly classified negative sentiment is “no problem.”

## Summary

In this analysis of Usenet messages, we’ve incorporated almost every method for tidy text mining described in this book, ranging from tf-idf to topic modeling, and from sentiment analysis to n-gram tokenization. Throughout the chapter, and indeed through all of our case studies, we’ve been able to rely on a small list of common tools for exploration and visualization. We hope that these examples show how much all tidy text analyses have in common with each other, and indeed with all tidy data analyses.