03 Analyzing word and document frequency: tf-idf

H. David Shea

30 Jul 2021

Contents

In these exercises, we are focused on determining "what a document is about". The approach is to look at *term frequency* (tf) of words in a document - trying to determine which words are "important" in the text. But some words - for instance, stop words - can be used frequently but may just occur naturally in high frequency. So, an alternative is to look at *inverse document frequency* (idf), where we decrease the weight for commonly used words and increase the weight of more infrequent words. (Frequency here is measured over collections of documents.)

"The statistic tf-idf is intended to measure how important a word is to a document in a collection (or corpus) of documents, for example, to one novel in a collection of novels or to one website in a collection of websites."

$$idf(\text{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

Term frequency in Jane Austen's novels

Word count per novel.

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

total_words <- book_words %>%
    group_by(book) %>%
    summarize(total = sum(n))

book_words <- left_join(book_words, total_words, by = "book")

book_words %>%
    slice_max(n, n = 10) %>%
    kable()
```

book	word	n	total
Mansfield Park	the	6206	160460
Mansfield Park	to	5475	160460
Mansfield Park	and	5438	160460
Emma	to	5239	160996
Emma	the	5201	160996
Emma	and	4896	160996
Mansfield Park	of	4778	160460
Pride & Prejudice	the	4331	122204
Emma	of	4291	160996
Pride & Prejudice	to	4162	122204

Term frequency per novel.

```
ggplot(book_words, aes(n / total, fill = book)) +
    geom_histogram(show.legend = FALSE, na.rm = TRUE) +
    xlim(NA, 0.0009) +
    facet_wrap( ~ book, ncol = 2, scales = "free_y") +
    theme_light()
```

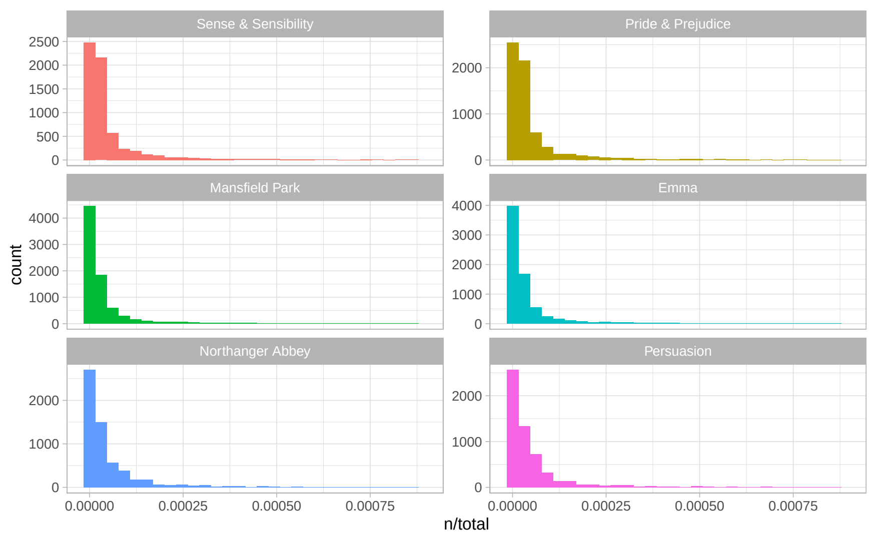


Figure 1: Term frequency distribution in Jane Austen's novels

Note: the distribution is similar across all novels - "many words that occur rarely and fewer words that occur frequently".

Zipf's law

"Zipf's law states that the frequency that a word appears is inversely proportional to its rank."

```
freq_by_rank <- book_words %>%
    group_by(book) %>%
    mutate(rank = row_number(),
           `term frequency` = n / total) %>%
```

```
    ungroup()

freq_by_rank %>%
    slice_max(n, n = 10) %>%
    kable()
```

book	word	n	total	rank	term frequency
Mansfield Park	the	6206	160460	1	0.0386763
Mansfield Park	to	5475	160460	2	0.0341207
Mansfield Park	and	5438	160460	3	0.0338901
Emma	to	5239	160996	1	0.0325412
Emma	the	5201	160996	2	0.0323052
Emma	and	4896	160996	3	0.0304107
Mansfield Park	of	4778	160460	4	0.0297769
Pride & Prejudice	the	4331	122204	1	0.0354407
Emma	of	4291	160996	4	0.0266528
Pride & Prejudice	to	4162	122204	2	0.0340578

We can visualize Zipf's law in a plot of term frequency versus rank.

```
freq_by_rank %>%
    ggplot(aes(rank, `term frequency`, color = book)) +
    geom_line(size = 1.1,
              alpha = 0.8,
              show.legend = FALSE) +
    scale_x_log10() +
    scale_y_log10()
```

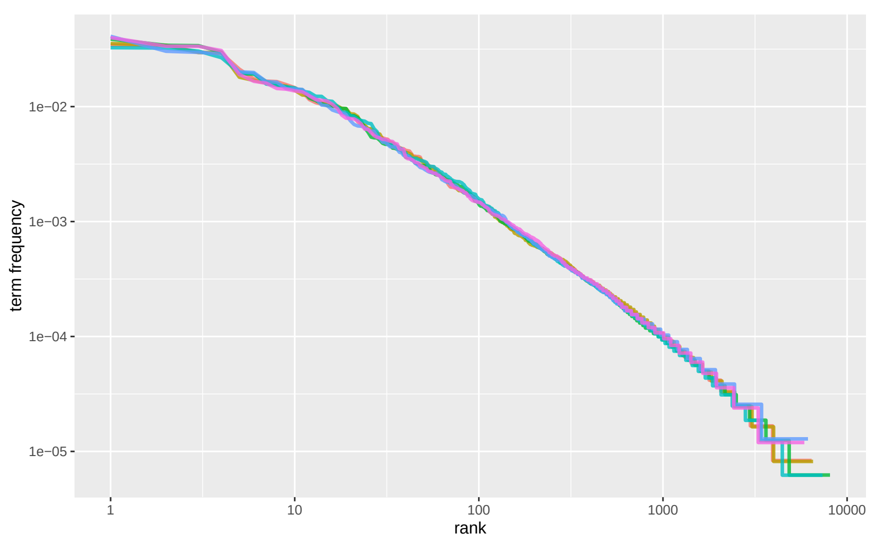
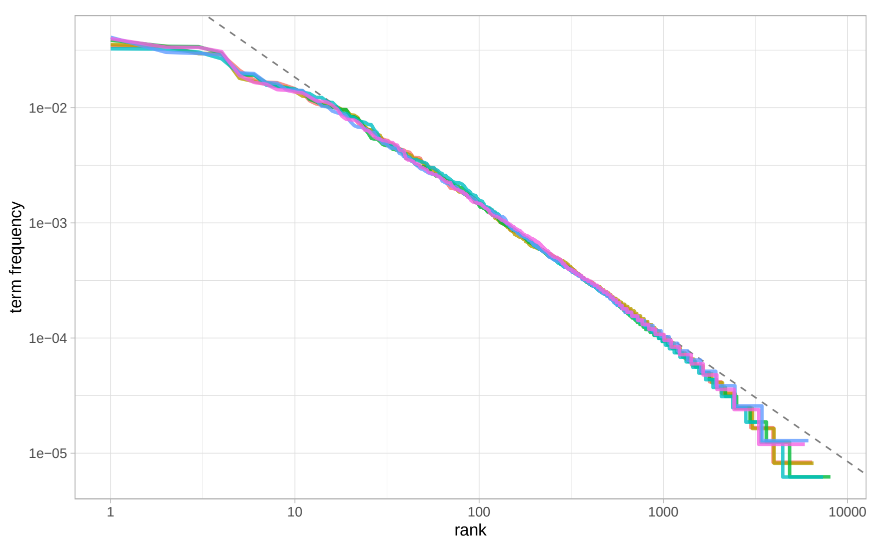


Figure 2: Zipf's law for Jane Austen's novels

In log-log scale, we can see that the novels are similar, but the negative slope is not constant - as you'd see if a power law relationship applied. But the "middle" of this distribution does seem more constant. And, indeed, thje slope there is close to `-1`.

```r
rank_subset <- freq_by_rank %>%
    filter(rank < 500,
           rank > 10)

rank_lm <- lm(log10(`term frequency`) ~ log10(rank), data = rank_subset)
rank_lm
#>
#> Call:
#> lm(formula = log10(`term frequency`) ~ log10(rank), data = rank_subset)
#>
#> Coefficients:
#> (Intercept)  log10(rank)
#>     -0.6226      -1.1125

freq_by_rank %>%
    ggplot(aes(rank, `term frequency`, color = book)) +
    geom_abline(
        intercept = rank_lm$coefficients[1],
        slope = rank_lm$coefficients[2],
        color = "gray50",
        linetype = 2
    ) +
    geom_line(size = 1.1,
              alpha = 0.8,
              show.legend = FALSE) +
    scale_x_log10() +
    scale_y_log10() +
    theme_light()
```



"The deviations we see here at high rank are not uncommon for many kinds of language; a corpus of language often contains fewer rare words than predicted by a single power law. The deviations at low rank are more unusual. Jane Austen uses a lower percentage of the most common words than many collections of language."

The `bind_tf_idf()` function

```
book_tf_idf <- book_words %>%
    bind_tf_idf(word, book, n)

book_tf_idf %>%
    slice_max(n, n = 10) %>%
    kable()
```

book	word	n	total	tf	idf	tf_idf
Mansfield Park	the	6206	160460	0.0386763	0	0
Mansfield Park	to	5475	160460	0.0341207	0	0
Mansfield Park	and	5438	160460	0.0338901	0	0
Emma	to	5239	160996	0.0325412	0	0
Emma	the	5201	160996	0.0323052	0	0
Emma	and	4896	160996	0.0304107	0	0
Mansfield Park	of	4778	160460	0.0297769	0	0
Pride & Prejudice	the	4331	122204	0.0354407	0	0
Emma	of	4291	160996	0.0266528	0	0
Pride & Prejudice	to	4162	122204	0.0340578	0	0

"Notice that idf and thus tf-idf are zero for these extremely common words. These are all words that appear in all six of Jane Austen's novels, so the idf term (which will then be the natural log of 1) is zero."

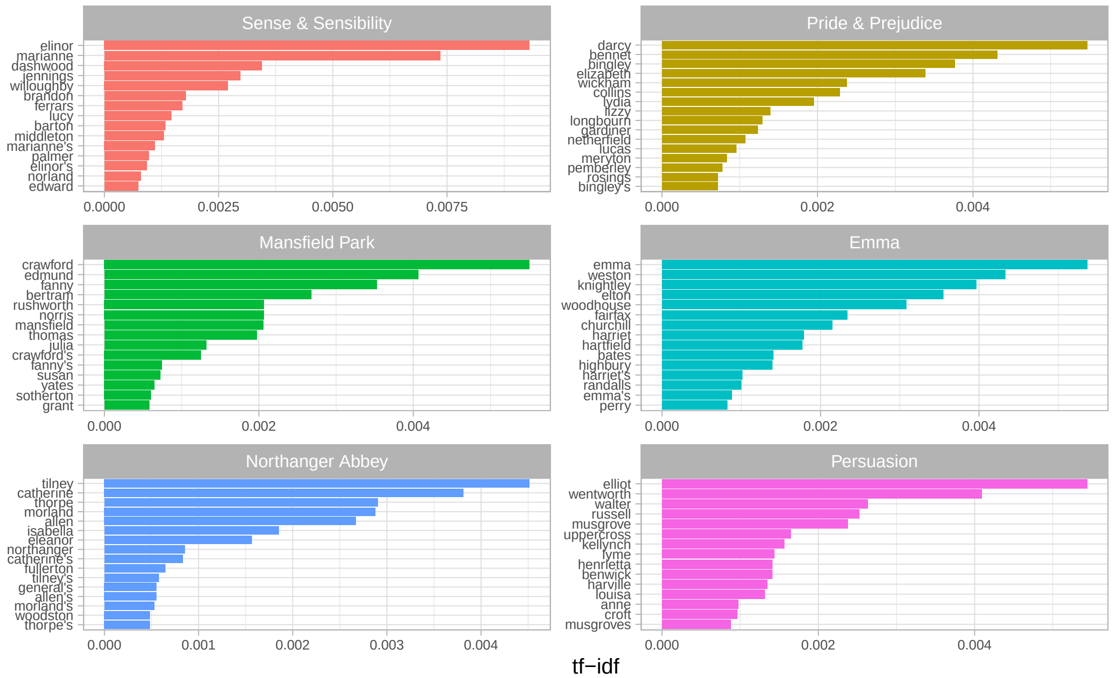
The high tf-idf words are shown like this.

```
book_tf_idf %>%
    select(-total) %>%
    arrange(desc(tf_idf)) %>%
    slice_max(tf_idf, n = 10) %>%
    kable()
```

book	word	n	tf	idf	tf_idf
Sense & Sensibility	elinor	623	0.0051935	1.791759	0.0093056
Sense & Sensibility	marianne	492	0.0041015	1.791759	0.0073488
Mansfield Park	crawford	493	0.0030724	1.791759	0.0055050
Pride & Prejudice	darcy	373	0.0030523	1.791759	0.0054689
Persuasion	elliot	254	0.0030362	1.791759	0.0054401
Emma	emma	786	0.0048821	1.098612	0.0053635
Northanger Abbey	tilney	196	0.0025199	1.791759	0.0045151
Emma	weston	389	0.0024162	1.791759	0.0043293
Pride & Prejudice	bennet	294	0.0024058	1.791759	0.0043106
Persuasion	wentworth	191	0.0022831	1.791759	0.0040908

```
book_tf_idf %>%
    group_by(book) %>%
    slice_max(tf_idf, n = 15) %>%
    ungroup() %>%
    ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = book)) +
```

```
    geom_col(show.legend = FALSE) +
    facet_wrap( ~ book, ncol = 2, scales = "free") +
    labs(x = "tf-idf", y = NULL) +
    theme_light() +
    theme(axis.text = element_text(size = 7))
```



"What measuring tf-idf has done here is show us that Jane Austen used similar language across her six novels, and what distinguishes one novel from the rest within the collection of her works are the proper nouns, the names of people and places. This is the point of tf-idf; it identifies words that are important to one document within a collection of documents."

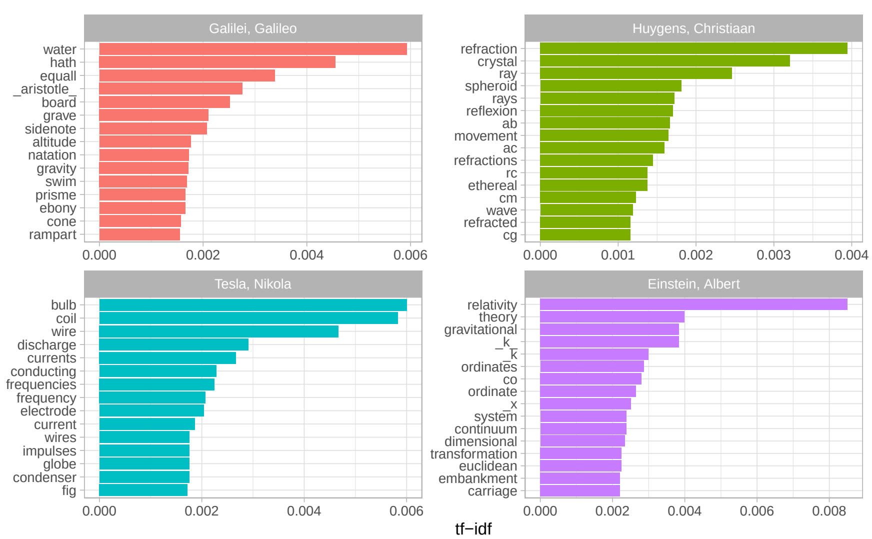
A corpus of physics texts

The following analyses uses some classis physics text available from *Project Gutenberg*. We will use *Discourse on Floating Bodies* by Galileo Galilei (ID = 37729), *Treatise on Light* by Christiaan Huygens (ID = 14725), *Experiments with Alternate Currents of High Potential and High Frequency* by Nikola Tesla (ID = 13476), and *Relativity: The Special and General Theory* by Albert Einstein (ID = 30155).

```
physics <- gutenberg_download(c(37729, 14725, 13476, 30155),
                              meta_fields = "author")

physics_words <- physics %>%
  unnest_tokens(word, text) %>%
  count(author, word, sort = TRUE)

physics_words %>%
    slice_max(n, n = 10) %>%
    kable()
```

author	word	n
Galilei, Galileo	the	3760
Tesla, Nikola	the	3604
Huygens, Christiaan	the	3553
Einstein, Albert	the	2993
Galilei, Galileo	of	2049
Einstein, Albert	of	2028
Tesla, Nikola	of	1737
Huygens, Christiaan	of	1708
Huygens, Christiaan	to	1207
Tesla, Nikola	a	1176

```
plot_physics <- physics_words %>%
    bind_tf_idf(word, author, n) %>%
    mutate(author = factor(
        author,
        levels = c(
            "Galilei, Galileo",
            "Huygens, Christiaan",
            "Tesla, Nikola",
            "Einstein, Albert"
        )
    ))

plot_physics %>%
    group_by(author) %>%
    slice_max(tf_idf, n = 15) %>%
    ungroup() %>%
    mutate(word = reorder(word, tf_idf)) %>%
    ggplot(aes(tf_idf, word, fill = author)) +
    geom_col(show.legend = FALSE) +
    labs(x = "tf-idf", y = NULL) +
    facet_wrap( ~ author, ncol = 2, scales = "free") +
    theme_light()
```



In the examples, we see some technical terms ('*k*', 'AB', "RC', etc.) that we might want to"clean up". And we see that the term 'co-ordinate' was broken up by the tokenizer into 'co' and 'ordinate'. We"clean up" these terms below.

```
physics %>%
    filter(str_detect(text, "_k_")) %>%
    select(text) %>%
    slice_sample(n = 10) %>%
    kable()
```

text
would needs be that from all the other points K_k_B there should
necessarily be equal to CD, because C_k_ is equal to CK, and C_g_ to
surface AB at the points AK_k_B. Then instead of the hemispherical
O_o_ has reached K_k_. Which is easy to comprehend, since, of these
the crystal at K_k_, all the points of the wave CO_oc_ will have
is the average density of the matter and *k* is a constant connected
CO_oc_ in the crystal, when O_o_ has arrived at K_k_, because it forms

```
physics %>%
    filter(str_detect(text, "RC")) %>%
    select(text) %>%
    slice_sample(n = 10) %>%
    kable()
```

text
degrees 40 minutes. Now let there be some other ray RC, the refraction
refraction of the ray RC.
be described, cutting the ray RC at R; and let RV be the perpendicular
incident rays. Let there be such a ray RC falling upon the surface
plane AFHE, the incident ray RC; it is required to find its refraction
that is to say, that the ray RC is refracted as CI.
explaining ordinary refraction. For the refraction of the ray RC is
the refraction of the ray RC, the proportion of CV to CD is 156,962 to
tangent of the complement of the angle RCV, which is 73 degrees 20
RC, the refraction of which it is required to find.

```
mystopwords <- tibble(
    word = c(
        "eq", "co", "rc", "ac", "ak",
        "bn", "fig", "file", "cg", "cb",
        "cm", "ab", "_k", "_k_", "_x"
    )
)

physics_words <- anti_join(physics_words, mystopwords,
                           by = "word")

plot_physics <- physics_words %>%
```

```
    bind_tf_idf(word, author, n) %>%
    mutate(word = str_remove_all(word, "_")) %>%
    group_by(author) %>%
    slice_max(tf_idf, n = 15) %>%
    ungroup() %>%
    mutate(word = reorder_within(word, tf_idf, author)) %>%
    mutate(author = factor(
        author,
        levels = c(
            "Galilei, Galileo",
            "Huygens, Christiaan",
            "Tesla, Nikola",
            "Einstein, Albert"
        )
    ))

ggplot(plot_physics, aes(word, tf_idf, fill = author)) +
    geom_col(show.legend = FALSE) +
    labs(x = NULL, y = "tf-idf") +
    facet_wrap( ~ author, ncol = 2, scales = "free") +
    coord_flip() +
    scale_x_reordered() +
    theme_light()
```