04 Relationships between words: n-grams and correlations

H. David Shea

30 Jul 2021

Contents

Now, we focus on relationships between words in a document - which words tend to occur together (immediate vicinity) or co-occur in the same document. An 'n-gram' is a token of "adjacent words".

Tokenizing by n-gram

Counting and filtering n-grams

```
austen_bigrams <- austen_books() %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)

austen_bigrams %>%
    slice_sample(n = 10) %>%
    kable()
```

book	bigram
Persuasion	and as
Pride & Prejudice	silent grave
Pride & Prejudice	to him
Mansfield Park	that it
Persuasion	affection could
Northanger Abbey	offered it
Emma	harriet's cheerful
Emma	delighted with
Mansfield Park	crawford was
Emma	of emma

```
austen_bigrams %>%
    count(bigram, sort = TRUE) %>%
    slice_max(n, n = 10) %>%
    kable()
```

bigram	n
NA	12242
of the	2853
to be	2670
in the	2221
it was	1691
i am	1485
she had	1405
of her	1363
to the	1315
she was	1309

```
bigrams_separated <- austen_bigrams %>%
    separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
    filter(!word1 %in% stop_words$word) %>%
    filter(!word2 %in% stop_words$word)

# new bigram counts:
bigram_counts <- bigrams_filtered %>%
    count(word1, word2, sort = TRUE)

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

word1	word2	n
NA	NA	12242
sir	thomas	266
miss	crawford	196
captain	wentworth	143
miss	woodhouse	143
frank	churchill	114
lady	russell	110
sir	walter	108
lady	bertram	101
miss	fairfax	98

```
bigrams_united <- bigrams_filtered %>%
    unite(bigram, word1, word2, sep = " ")

bigrams_united %>%
    slice_sample(n = 10) %>%
    kable()
```

book	bigram
Mansfield Park	means desirable

book	bigram
Mansfield Park	NA NA
Emma	spring precisely
Sense & Sensibility	materially altered
Emma	mixed feelings
Northanger Abbey	NA NA
Northanger Abbey	altogether pleased
Emma	bringing forward
Pride & Prejudice	recollections obtruded
Emma	hair cut

```
austen_books() %>%
    unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
    separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
    filter(!word1 %in% stop_words$word,!word2 %in% stop_words$word,!word3 %in% stop_words$word) %>%
    count(word1, word2, word3, sort = TRUE) %>%
    slice_max(n, n = 10) %>%
    kable()
```

word1	word2	word3	n
NA	NA	NA	13260
dear	miss	woodhouse	20
miss	de	bourgh	17
lady	catherine	de	11
poor	miss	taylor	11
sir	walter	elliot	10
catherine	de	bourgh	9
dear	sir	thomas	8
replied	miss	crawford	7
sir	william	lucas	7
ten	thousand	pounds	7

Analyzing bigrams

```
bigrams_filtered %>%
  filter(word2 == "street") %>%
  count(book, word1, sort = TRUE) %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Most Common 'Street' Names in Jane Austen's Novels")
```

Table 6: Most Common 'Street' Names in Jane Austen's Novels

book	word1	n
Sense & Sensibility	harley	16
Sense & Sensibility	berkeley	15
Northanger Abbey	milsom	10
Northanger Abbey	pulteney	10

book	word1	n
Mansfield Park	wimpole	9
Pride & Prejudice	gracechurch	8
Persuasion	milsom	5
Sense & Sensibility	bond	4
Sense & Sensibility	conduit	4
Persuasion	rivers	4

```
bigram_tf_idf <- bigrams_united %>%
  count(book, bigram) %>%
  bind_tf_idf(bigram, book, n) %>%
  arrange(desc(tf_idf))

bigram_tf_idf %>%
    slice_max(tf_idf, n = 10) %>%
    kable(caption = "tf-idf or Bigrams in Jane Austen's Novels")
```

Table 7: tf-idf or Bigrams in Jane Austen's Novels

book	bigram	n	tf	idf	tf_idf
Mansfield Park	sir thomas	266	0.0244238	1.79176	0.0437616
Persuasion	captain wentworth	143	0.0232143	1.79176	0.0415944
Mansfield Park	miss crawford	196	0.0179965	1.79176	0.0322454
Persuasion	lady russell	110	0.0178571	1.79176	0.0319957
Persuasion	sir walter	108	0.0175325	1.79176	0.0314140
Emma	miss woodhouse	143	0.0128817	1.79176	0.0230809
Northanger Abbey	miss tilney	74	0.0127829	1.79176	0.0229038
Sense & Sensibility	colonel brandon	96	0.0114572	1.79176	0.0205286
Sense & Sensibility	sir john	94	0.0112185	1.79176	0.0201009
Emma	frank churchill	114	0.0102693	1.79176	0.0184002

```
bigram_tf_idf %>%
    group_by(book) %>%
    slice_max(tf_idf, n = 15) %>%
    ungroup() %>%
    ggplot(aes(tf_idf, fct_reorder(bigram, 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))
```

"There are advantages and disadvantages to examining the tf-idf of bigrams rather than individual words. Pairs of consecutive words might capture structure that isn't present when one is just counting single words, and may provide context that makes tokens more understandable."

"However, the per-bigram counts are also sparser: a typical two-word pair is rarer than either of its component words. Thus, bigrams can be especially useful when you have a very large text dataset."

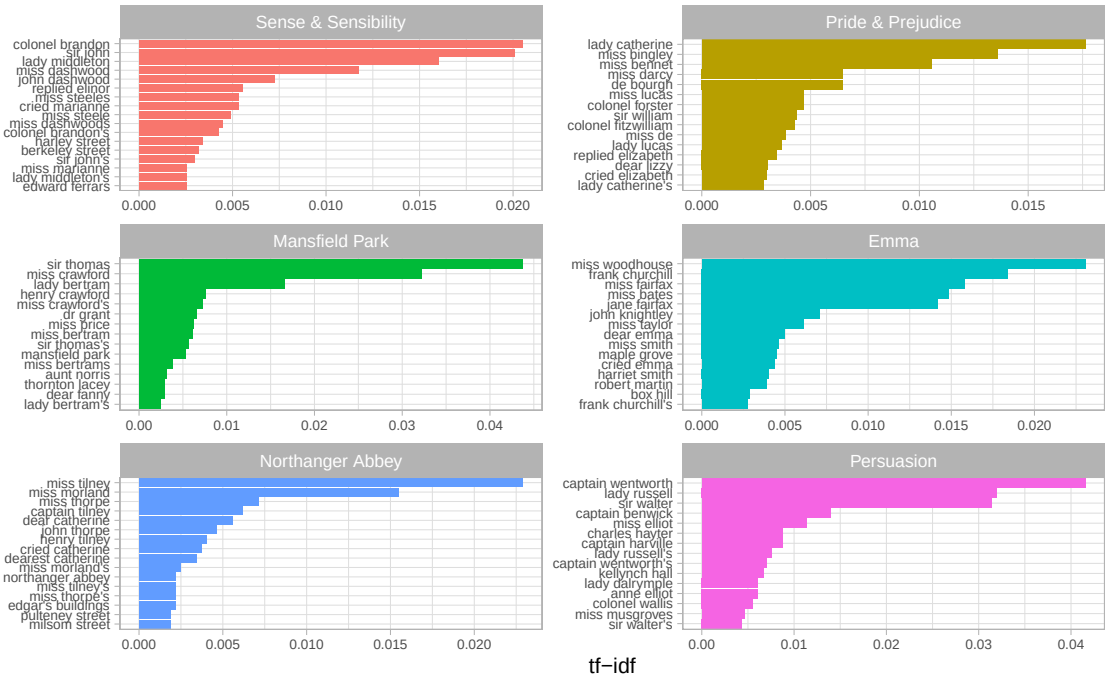


Figure 1: Bigrams with the highest tf-idf from each Jane Austen novel

Using bigrams to provide context in sentiment analysis

Bigrams can help get at true context for terms like 'not happy', for instance.

```
bigrams_separated %>%
    filter(word1 == "not") %>%
    count(word1, word2, sort = TRUE) %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Words preceeded by 'not' in in Jane Austen's Novels")
```

Table 8: Words preceeded by 'not' in in Jane Austen's Novels

word1	word2	n
not	be	580
not	to	335
not	have	307
not	know	237
not	a	184
not	think	162
not	been	151
not	the	135
not	at	126
not	in	110

```
AFINN <- get_sentiments("afinn")
```

```
not_words <- bigrams_separated %>%
    filter(word1 == "not") %>%
    inner_join(AFINN, by = c(word2 = "word")) %>%
    count(word2, value, sort = TRUE)

not_words %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Sentiment value of words preceeded by 'not' in in Jane Austen's Novels")
```

Table 9: Sentiment value of words preceeded by 'not' in in Jane Austen's Novels

word2	value	n
like	2	95
help	2	77
want	1	41
wish	1	39
allow	1	30
care	2	21
sorry	-1	20
leave	-1	17
pretend	-1	17
worth	2	17

We can calculater (and visualize) the sentiment impact of words that contribute in the 'wrong' direction to sentiment. Here `contribution` is the sentiment value of the word times the number of times the word appeared in the text preceded by 'not'.

```
not_words %>%
    mutate(contribution = n * value) %>%
    arrange(desc(abs(contribution))) %>%
    head(20) %>%
    mutate(word2 = reorder(word2, contribution)) %>%
    ggplot(aes(n * value, word2, fill = n * value > 0)) +
    geom_col(show.legend = FALSE) +
    labs(x = "Sentiment value * number of occurrences",
         y = "Words preceded by \"not\"") +
    theme_light()
```

We can expand to more 'negated words' beyond just 'not'.

```
negation_words <- c("not", "no", "never", "without")

negated_words <- bigrams_separated %>%
    filter(word1 %in% negation_words) %>%
    inner_join(AFINN, by = c(word2 = "word")) %>%
    count(word1, word2, value, sort = TRUE)

negated_words %>%
    mutate(contribution = n * value,
           word2 = reorder(paste(word2, word1, sep = "__"), contribution)) %>%
```

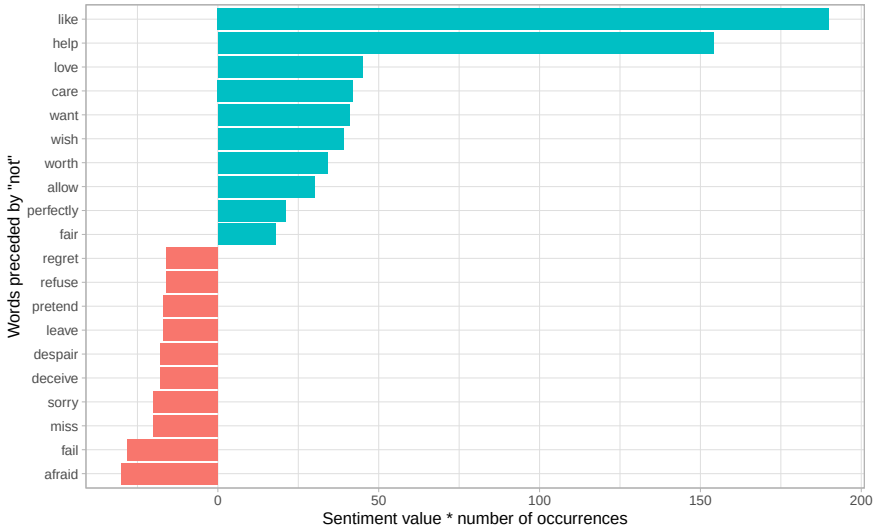


Figure 2: Words preceded by 'not' that had the greatest contribution to sentiment values, in either a positive or negative direction

```
    group_by(word1) %>%
    slice_max(abs(contribution), n = 12, with_ties = FALSE) %>%
    ggplot(aes(word2, contribution, fill = n * value > 0)) +
    geom_col(show.legend = FALSE) +
    facet_wrap( ~ word1, scales = "free") +
    scale_x_discrete(
        labels = function(x)
            gsub("__.+$", "", x)
    ) +
    xlab("Words preceded by negation term") +
    ylab("Sentiment value * # of occurrences") +
    coord_flip() +
    theme_light()
```

Visualizing a network of bigrams with ggraph

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

word1	word2	n
NA	NA	12242
sir	thomas	266
miss	crawford	196
captain	wentworth	143
miss	woodhouse	143
frank	churchill	114
lady	russell	110

word1	word2	n
sir	walter	108
lady	bertram	101
miss	fairfax	98

```
bigram_graph <- bigram_counts %>%
    filter(n > 20) %>%
    graph_from_data_frame()
#> Warning in graph_from_data_frame(.): In `d' `NA' elements were replaced with
#> string "NA"

bigram_graph
#> IGRAPH 5a3a5d6 DN-- 86 71 --
#> + attr: name (v/c), n (e/n)
#> + edges from 5a3a5d6 (vertex names):
#>  [1] NA      ->NA         sir     ->thomas     miss    ->crawford
#>  [4] captain ->wentworth  miss    ->woodhouse  frank   ->churchill
#>  [7] lady    ->russell    sir     ->walter     lady    ->bertram
#> [10] miss    ->fairfax    colonel ->brandon    sir     ->john
#> [13] miss    ->bates      jane    ->fairfax    lady    ->catherine
#> [16] lady    ->middleton  miss    ->tilney     miss    ->bingley
#> [19] thousand->pounds     miss    ->dashwood   dear    ->miss
#> [22] miss    ->bennet     miss    ->morland    captain ->benwick
#> + ... omitted several edges

set.seed(2017)

ggraph(bigram_graph, layout = "fr") +
    geom_edge_link() +
    geom_node_point() +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

Note in the graph that proper title ("miss", "sir", etc.) are common centers of nodes. We also see common short phrases like 'half hour' and 'maple grove'.

```
set.seed(2020)

a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

ggraph(bigram_graph, layout = "fr") +
    geom_edge_link(
        aes(edge_alpha = n),
        show.legend = FALSE,
        arrow = a,
        end_cap = circle(.07, 'inches')
    ) +
    geom_node_point(color = "lightblue", size = 5) +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
    theme_void()
```

"Note that this is a visualization of a Markov chain, a common model in text processing. In a Markov chain, each choice of word depends only on the previous word. In this case, a random generator following this model

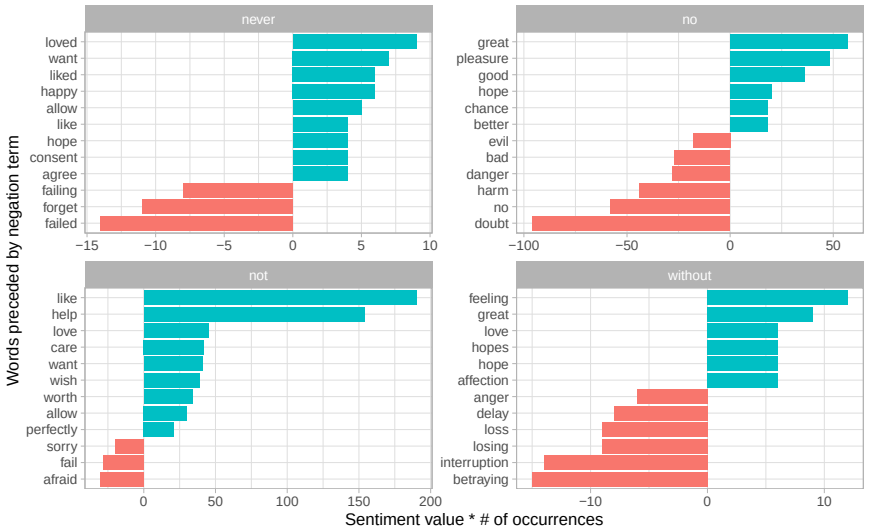


Figure 3: Most common positive or negative words to follow negations such as 'never', 'no', 'not', and 'without'

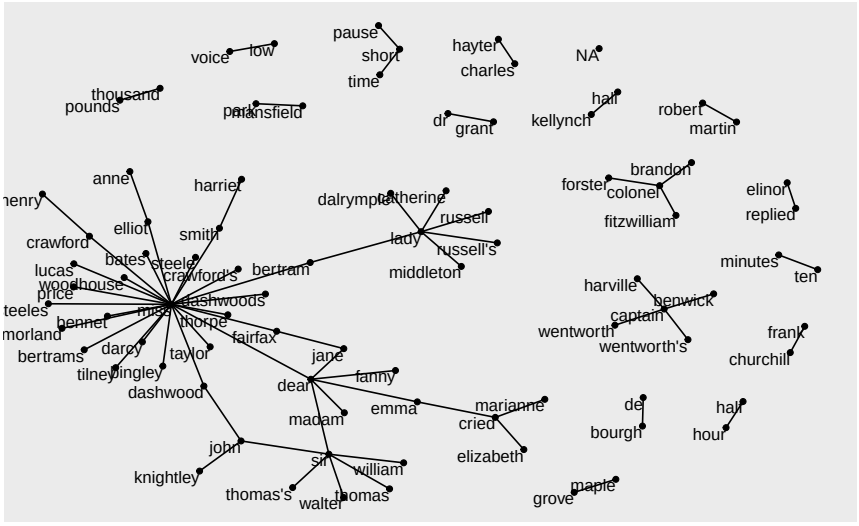


Figure 4: Common bigrams in Jane Austen's novels, showing those that occurred more than 20 times and where neither word was a stop word

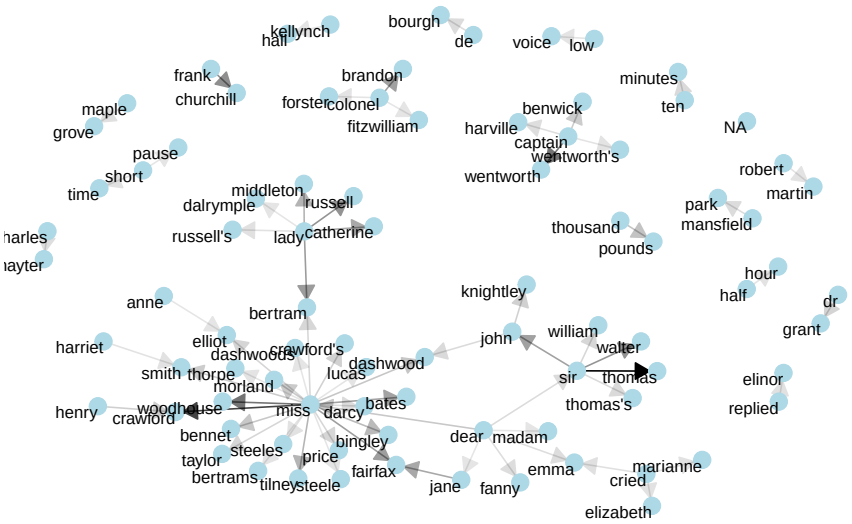


Figure 5: Common bigrams in Jane Austen's novels, with some polishing

might spit out "dear", then "sir", then "william/walter/thomas/thomas's", by following each word to the most common words that follow it. To make the visualization interpretable, we chose to show only the most common word to word connections, but one could imagine an enormous graph representing all connections that occur in the text."

Visualizing bigrams in other texts

First, some useful shortcut functions.

```
count_bigrams <- function(dataset, .stop_words = stop_words) {
    dataset %>%
        unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
        separate(bigram, c("word1", "word2"), sep = " ") %>%
        filter(!word1 %in% .stop_words$word,!word2 %in% .stop_words$word) %>%
        count(word1, word2, sort = TRUE)
}

visualize_bigrams <- function(bigrams) {
    set.seed(2016)
    a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

    bigrams %>%
        graph_from_data_frame() %>%
        ggraph(layout = "fr") +
        geom_edge_link(aes(edge_alpha = n),
                       show.legend = FALSE,
                       arrow = a) +
        geom_node_point(color = "lightblue", size = 5) +
        geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
        theme_void()
}
```

Visualizing the bigrams in the King James Version of the bible (ID = 10) from *Project Gutenberg*.

```
# the King James version is book 10 on Project Gutenberg:
kjv <- gutenberg_download(10)

kjv_bigrams <- kjv %>%
    count_bigrams()

# filter out rare combinations, as well as digits
kjv_bigrams %>%
    filter(n > 40,!str_detect(word1, "\\d"),!str_detect(word2, "\\d")) %>%
    visualize_bigrams()
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'father's' in 'mbcsToSbcs': dot substituted for <e2>
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'father's' in 'mbcsToSbcs': dot substituted for <80>
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'father's' in 'mbcsToSbcs': dot substituted for <99>
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'king's' in 'mbcsToSbcs': dot substituted for <e2>
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'king's' in 'mbcsToSbcs': dot substituted for <80>
#> Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
#> conversion failure on 'king's' in 'mbcsToSbcs': dot substituted for <99>
```

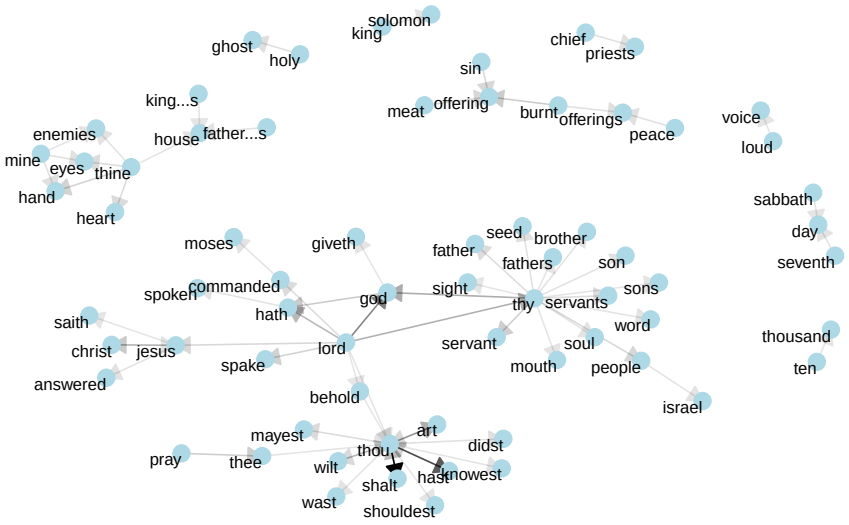


Figure 6: Directed graph of common bigrams in the King James Bible, showing those that occurred more than 40 times

Counting and correlating pairs of words with the widyr package

We can also look at words that tend to occur together within documents or parts of documents, whether they are next to each other or not.

Counting and correlating among sections

We look at *Pride and Prejudice* divided into 10-line sections.

```
austen_section_words <- austen_books() %>%
    filter(book == "Pride & Prejudice") %>%
    mutate(section = row_number() %/% 10) %>%
    filter(section > 0) %>%
    unnest_tokens(word, text) %>%
    filter(!word %in% stop_words$word)

austen_section_words %>%
    slice_sample(n = 10) %>%
    kable(caption = "10-Line Sections from _Pride & Prejudice_")
```

Table 11: 10-Line Sections from *Pride & Prejudice*

book	section	word
Pride & Prejudice	361	oppose
Pride & Prejudice	11	twenty
Pride & Prejudice	354	quarter
Pride & Prejudice	1212	lizzy
Pride & Prejudice	605	glancing
Pride & Prejudice	930	amusement
Pride & Prejudice	462	idea
Pride & Prejudice	1108	bingley
Pride & Prejudice	168	giving
Pride & Prejudice	77	plain

```
# count words co-occuring within sections
word_pairs <- austen_section_words %>%
    pairwise_count(word, section, sort = TRUE)
#> Warning: `distinct_()` was deprecated in dplyr 0.7.0.
#> Please use `distinct()` instead.
#> See vignette('programming') for more help
#> Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
#> Please use `tibble::as_tibble()` instead.

word_pairs %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Word Pairs per 10-Line Section in _Pride & Prejudice_")
```

Table 12: Word Pairs per 10-Line Section in *Pride & Prejudice*

item1	item2	n
darcy	elizabeth	144
elizabeth	darcy	144
miss	elizabeth	110
elizabeth	miss	110
elizabeth	jane	106
jane	elizabeth	106
miss	darcy	92
darcy	miss	92

item1	item2	n
elizabeth	bingley	91
bingley	elizabeth	91

```
word_pairs %>%
  filter(item1 == "darcy") %>%
    slice_max(n, n = 10) %>%
    kable(caption = "Word that Co-Occur with 'Darcy' per 10-Line Section in _Pride & Prejudice_")
```

Table 13: Word that Co-Occur with 'Darcy' per 10-Line Section in *Pride & Prejudice*

item1	item2	n
darcy	elizabeth	144
darcy	miss	92
darcy	bingley	86
darcy	jane	46
darcy	bennet	45
darcy	sister	45
darcy	time	41
darcy	lady	38
darcy	friend	37
darcy	wickham	37

Pairwise correlation

```
# we need to filter for at least relatively common words first
word_cors <- austen_section_words %>%
    group_by(word) %>%
    filter(n() >= 20) %>%
    pairwise_cor(word, section, sort = TRUE)

word_cors %>%
    slice_max(correlation, n = 10) %>%
    kable(caption = "Word Pair Correlation per 10-Line Section in _Pride & Prejudice_")
```

Table 14: Word Pair Correlation per 10-Line Section in *Pride & Prejudice*

item1	item2	correlation
bourgh	de	0.9508501
de	bourgh	0.9508501
pounds	thousand	0.7005808
thousand	pounds	0.7005808
william	sir	0.6644719
sir	william	0.6644719
catherine	lady	0.6633048

item1	item2	correlation
lady	catherine	0.6633048
forster	colonel	0.6220950
colonel	forster	0.6220950

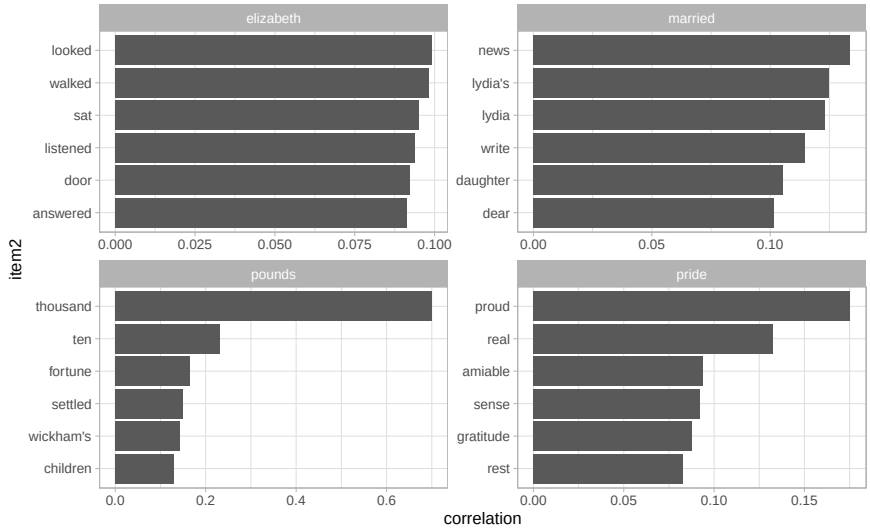
```
word_cors %>%
  filter(item1 == "pounds") %>%
    slice_max(correlation, n = 10) %>%
    kable(caption = "Word Pair - including 'pounds' - Correlation per 10-Line Section in _Pride & Preju
```

Table 15: Word Pair - including 'pounds' - Correlation per 10-Line Section in *Pride & Prejudice*

item1	item2	correlation
pounds	thousand	0.7005808
pounds	ten	0.2305758
pounds	fortune	0.1638626
pounds	settled	0.1494605
pounds	wickham's	0.1415240
pounds	children	0.1290001
pounds	mother's	0.1190593
pounds	believed	0.0932152
pounds	estate	0.0889688
pounds	ready	0.0859704
pounds	particulars	0.0859704

```
word_cors %>%
    filter(item1 %in% c("elizabeth", "pounds", "married", "pride")) %>%
    group_by(item1) %>%
    slice_max(correlation, n = 6) %>%
    ungroup() %>%
    mutate(item2 = reorder(item2, correlation)) %>%
    ggplot(aes(item2, correlation)) +
    geom_bar(stat = "identity") +
    facet_wrap( ~ item1, scales = "free") +
    coord_flip() +
    theme_light()
```

\begin{figure}



{

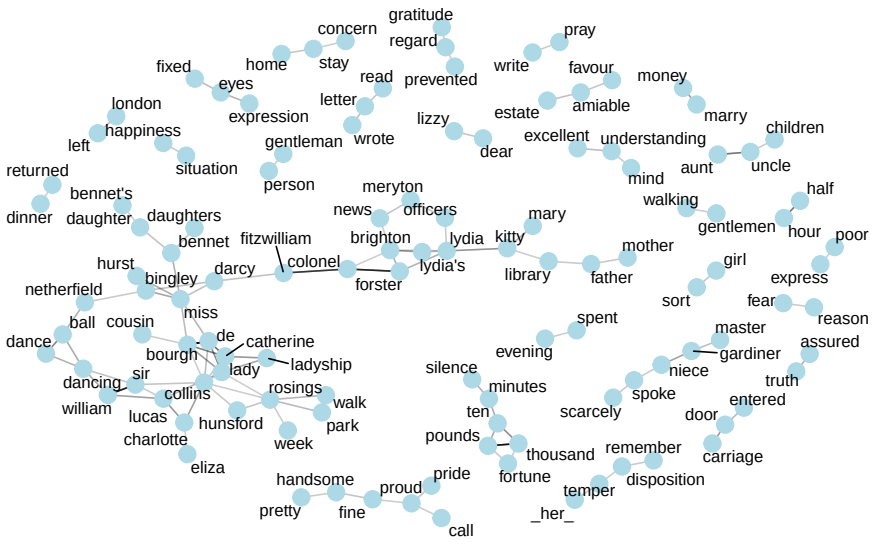
}

\caption{Words from *Pride and Prejudice* that were most correlated with 'elizabeth', 'pounds', 'married', and 'pride'} \end{figure}

```
set.seed(2016)

word_cors %>%
    filter(correlation > .15) %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
    geom_node_point(color = "lightblue", size = 5) +
    geom_node_text(aes(label = name), repel = TRUE) +
    theme_void()
```

\begin{figure}



{

}

\caption{Pairs of words in *Pride and Prejudice* that show at least a .15 correlation of appearing within the same 10-line section} \end{figure}