Sentiment Analysis

SDS 322E
October 17, 2025

Dr. Lydia R. Lucchesi
Department of Statistics and Data Sciences
The University of Texas at Austin

Week 8

	Monday	Tuesday	Wednesday	Thursday	Friday
Lecture	Project 1 Workday		Introduction to Tidy Text		Sentiment Analysis
Other	Lab 6	Office Hours (3-5PM)	Office Hours (3-5PM)	Project 1 Due	Pre-Lab 7 Quiz Due

In text analysis, what do we call a meaningful unit of text (most often a word)?

Which function from the **tidytext** R package do we use to tokenize text data?

a. `tokenize_text()`
b. `split_text()`
c. `unnest_tokens()`

Which of the three lexicons in the **stop_words** tibble contains the most stop words?

a. onix
b. SMART
c. snowball

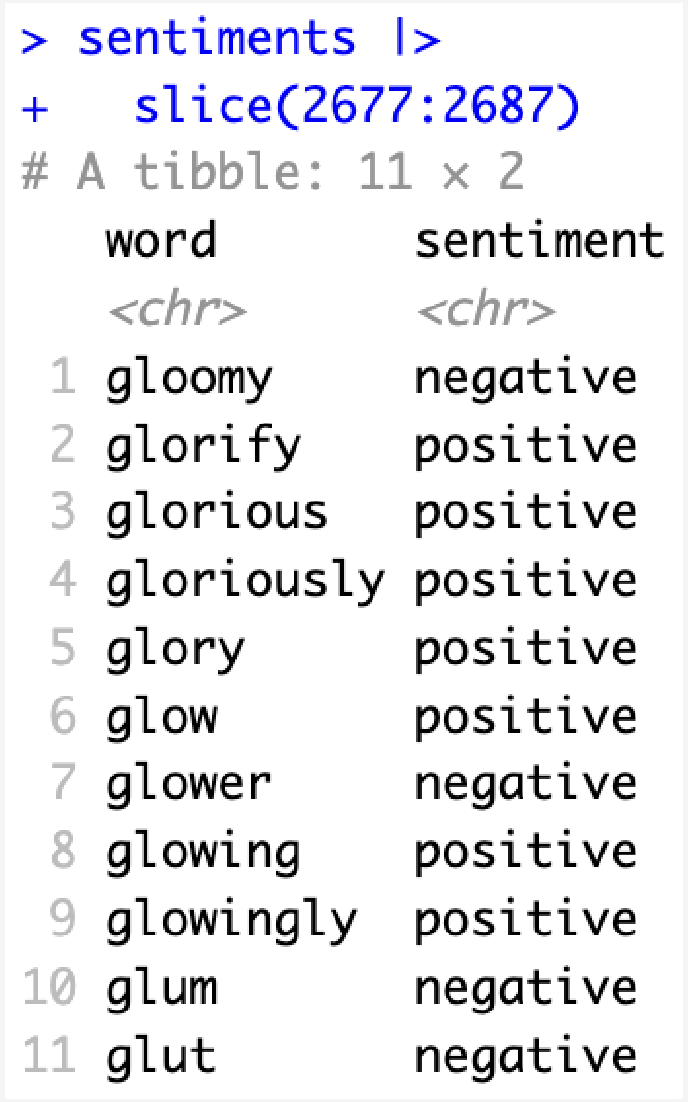
Positive or negative sentiment?

When the plane landed, I could see from my little window that the weather was gloomy. I trudged over to baggage claim, where I waited for many minutes, before it became frustratingly clear that my luggage had been lost.

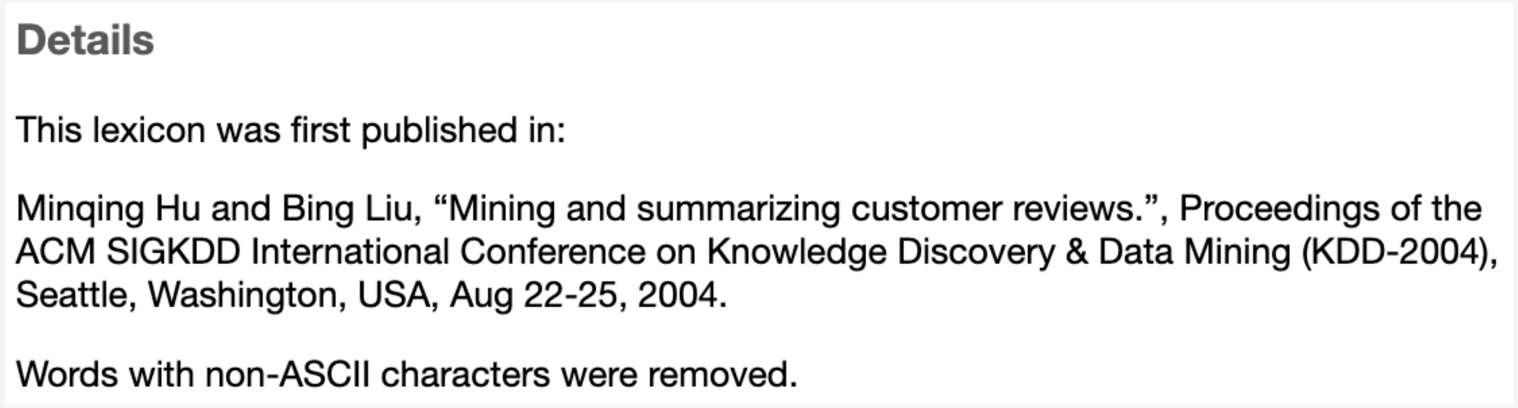
Positive or negative sentiment?

The landing was smooth, and when the plane parked at the gate, I could barely contain my excitement. My favorite place in the world was beckoning to me.

The **sentiments** dataset in the **tidytext** R package



```
> sentiments |>
+   slice(2677:2687)
# A tibble: 11 × 2
   word       sentiment
   <chr>      <chr>
 1 gloomy     negative
 2 glorify    positive
 3 glorious   positive
 4 gloriously positive
 5 glory      positive
 6 glow       positive
 7 glower     negative
 8 glowing    positive
 9 glowingly  positive
10 glum       negative
11 glut       negative
```

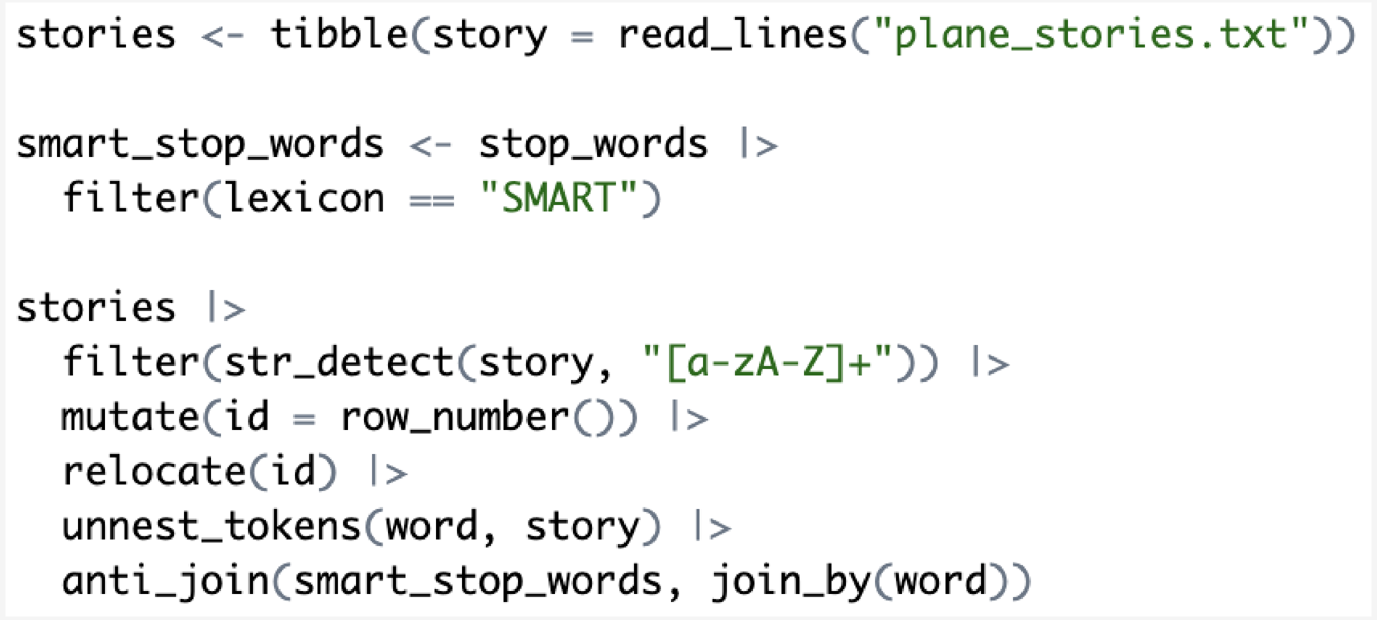


Details

This lexicon was first published in:

Minqing Hu and Bing Liu, “Mining and summarizing customer reviews.”, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004), Seattle, Washington, USA, Aug 22-25, 2004.

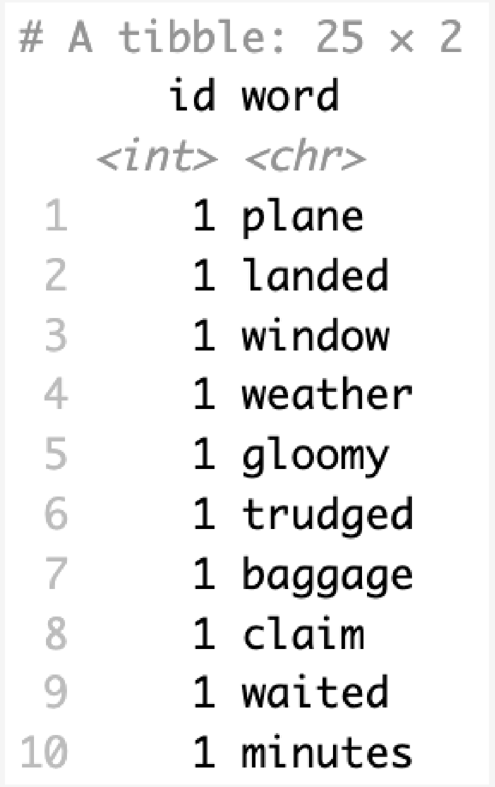
Words with non-ASCII characters were removed.



```
stories <- tibble(story = read_lines("plane_stories.txt"))

smart_stop_words <- stop_words |>
  filter(lexicon == "SMART")

stories |>
  filter(str_detect(story, "[a-zA-Z]+")) |>
  mutate(id = row_number()) |>
  relocate(id) |>
  unnest_tokens(word, story) |>
  anti_join(smart_stop_words, join_by(word))
```



```
# A tibble: 25 × 2
      id word
   <int> <chr>
 1     1 plane
 2     1 landed
 3     1 window
 4     1 weather
 5     1 gloomy
 6     1 trudged
 7     1 baggage
 8     1 claim
 9     1 waited
10     1 minutes
```

The next step is to add on sentiment information. If we want to keep only words that exist in both **stories** and **sentiments**, which join function should we use?

```r
stories <- tibble(story = read_lines("plane_stories.txt"))

smart_stop_words <- stop_words |>
  filter(lexicon == "SMART")

stories |>
  filter(str_detect(story, "[a-zA-Z]+")) |>
  mutate(id = row_number()) |>
  relocate(id) |>
  unnest_tokens(word, story) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all")
```

```
# A tibble: 8 × 3
     id word          sentiment
  <int> <chr>         <chr>
1     1 gloomy        negative   🔴
2     1 frustratingly negative   🔴
3     1 clear         positive   🟢
4     1 lost          negative   🔴
5     2 smooth        positive   🟢
6     2 excitement    positive   🟢
7     2 favorite      positive   🟢
8     2 beckoning     positive   🟢
```

```r
sentiments |>
  count(word) |>
  filter(n > 1)
```

```
# A tibble: 3 × 2
  word            n
  <chr>       <int>
1 envious         2
2 enviously       2
3 enviousness     2
```

Analyzing sentiment in texts from Project Gutenberg

```r
library(tidyverse)
library(tidytext)
# So we can download public domain works from Project Gutenberg
library(gutenbergr)

# Get SMART stop word lexicon
smart_stop_words <- stop_words |>
  filter(lexicon == "SMART")

# Download the book Frankenstein
book <- gutenberg_download(84)

# Tidy and tokenize, remove stop words, and add sentiment
book |>
  mutate(chapter = cumsum(str_detect(text, "^Chapter"))) |>
  filter(chapter > 0) |>
  filter(!str_detect(text, "^Chapter")) |>
  filter(str_detect(text, "[a-zA-Z]+")) |>
  unnest_tokens(word, text) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all")
```

Analyzing sentiment in texts from Project Gutenberg

```r
book |>
  mutate(chapter = cumsum(str_detect(text, "^Chapter"))) |>
  filter(chapter > 0) |>
  filter(!str_detect(text, "^Chapter")) |>
  filter(str_detect(text, "[a-zA-Z]+")) |>
  unnest_tokens(word, text) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all") |>
  group_by(chapter, sentiment) |>
  summarize(n = n(), .groups = "drop") |>
  pivot_wider(names_from = "sentiment", values_from = "n") |>
  mutate(proportion = positive / (positive + negative))
```

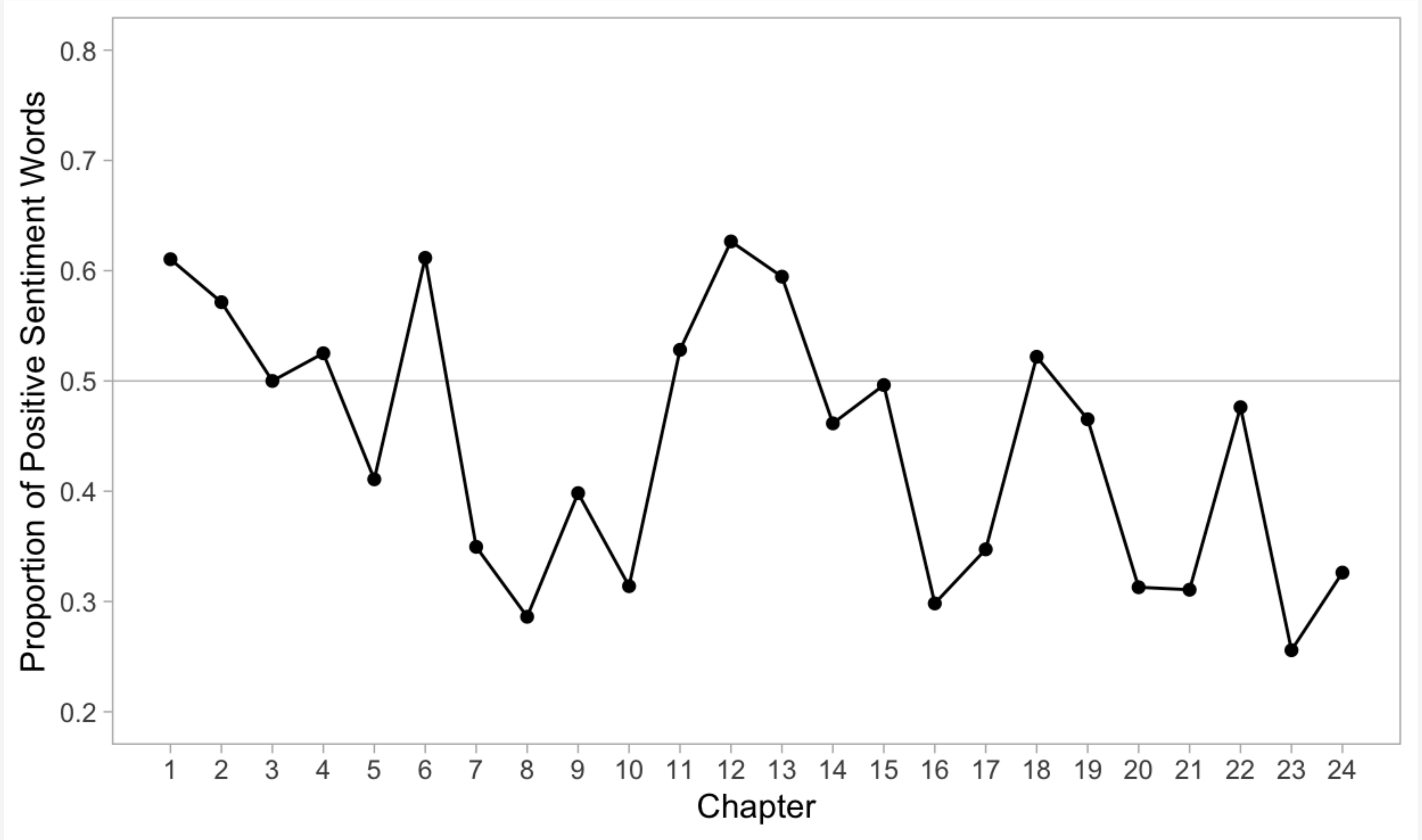
Analyzing sentiment in texts from Project Gutenberg

```r
book |>
  mutate(chapter = cumsum(str_detect(text, "^Chapter"))) |>
  filter(chapter > 0) |>
  filter(!str_detect(text, "^Chapter")) |>
  filter(str_detect(text, "[a-zA-Z]+")) |>
  unnest_tokens(word, text) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all") |>
  group_by(chapter, sentiment) |>
  summarize(n = n(), .groups = "drop") |>
  pivot_wider(names_from = "sentiment", values_from = "n") |>
  mutate(proportion = positive / (positive + negative))
```

```
# A tibble: 24 × 4
   chapter negative positive proportion
     <int>    <int>    <int>      <dbl>
 1       1       60       94      0.610
 2       2       84      112      0.571
 3       3       88       88      0.5
 4       4      104      115      0.525
 5       5      109       76      0.411
 6       6       80      126      0.612
 7       7      214      115      0.350
 8       8      227       91      0.286
 9       9      139       92      0.398
10      10      153       70      0.314
```

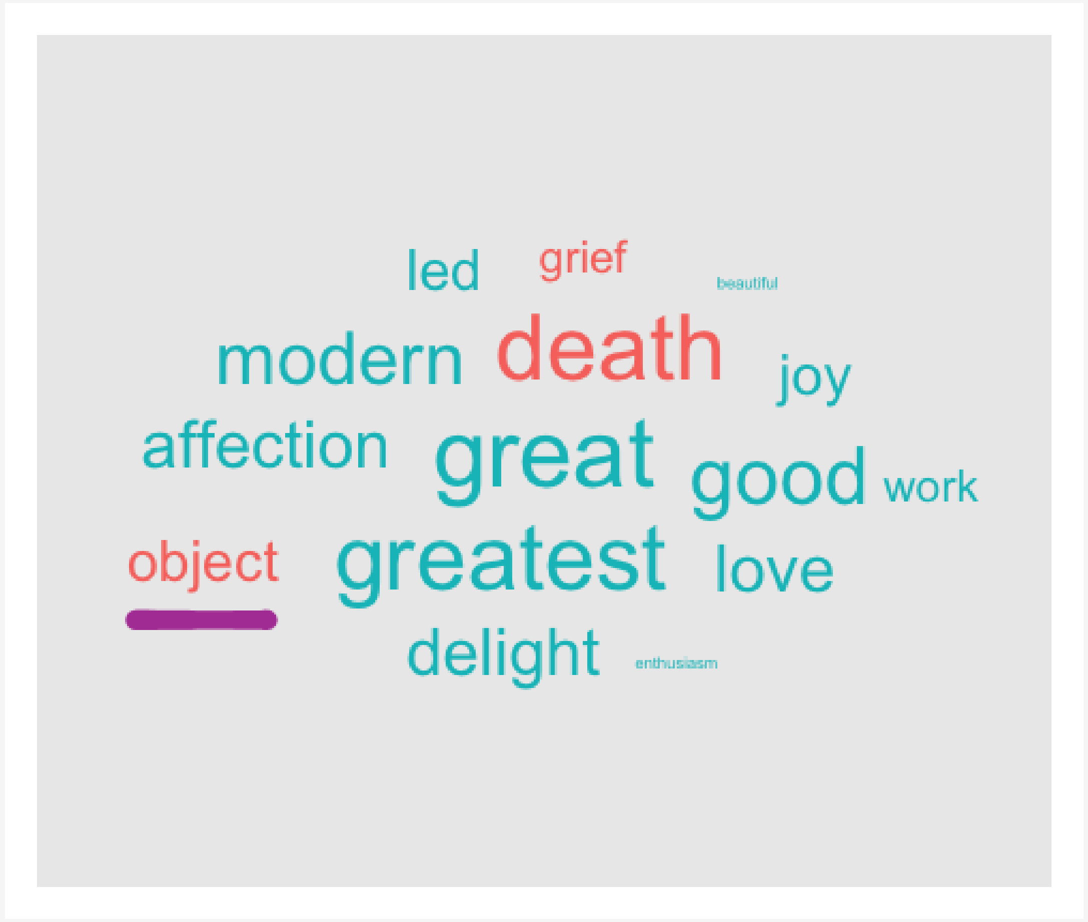
Analyzing sentiment in texts from Project Gutenberg

```r
book |>
  mutate(chapter = cumsum(str_detect(text, "^Chapter"))) |>
  filter(chapter > 0) |>
  filter(!str_detect(text, "^Chapter")) |>
  filter(str_detect(text, "[a-zA-Z]+")) |>
  unnest_tokens(word, text) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all") |>
  group_by(chapter, sentiment) |>
  summarize(n = n(), .groups = "drop") |>
  pivot_wider(names_from = "sentiment", values_from = "n") |>
  mutate(proportion = positive / (positive + negative)) |>
  ggplot(aes(x = chapter, y = proportion)) +
  geom_hline(yintercept = .5, linewidth = .25, color = "gray") +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:24) +
  scale_y_continuous(limits = c(0.2, 0.8),
                     breaks = seq(0.2, 0.8, 0.1)) +
  theme_light() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_blank()) +
  labs(x = "Chapter",
       y = "Proportion of Positive Sentiment Words")
```



ggwordcloud R package

```
book |>
  mutate(chapter = cumsum(str_detect(text, "^Chapter"))) |>
  filter(chapter > 0) |>
  filter(!str_detect(text, "^Chapter")) |>
  filter(str_detect(text, "[a-zA-Z]+")) |>
  unnest_tokens(word, text) |>
  anti_join(smart_stop_words, join_by(word)) |>
  inner_join(sentiments, join_by(word), multiple = "all") |>
  filter(chapter %in% 1:5) |>
  group_by(word, sentiment) |>
  summarize(n = n(), .groups = "drop") |>
  arrange(desc(n)) |>
  slice(1:15) |>
  ggplot(aes(label = word, size = n, color = sentiment)) +
  geom_text_wordcloud()
```


led
grief
beautiful
modern
death
joy
affection
great
good
work
object
greatest
love
delight
enthusiasm