An Introduction to Text Mining using R

Dr Austin R Brown

Kennesaw State University

Introduction

- When we think of statistics and data science, a lot of times we think about quantitative data.
- However, more and more, rich sources of useful data take the form of unstructrured strings of text.
- For instance, customer reviews or student evaluations or reflections are really useful, informative pieces of information.
  - But these are not numbers. These are strings of text!

Introduction

- There are a growing number of ways to analyze these types of data including things like natural language processing (NLP) and large language models (LLMs).
- Another way of exploring and analyzing textual data is through a general method called *text mining*.
- I like to think of text mining as a "quick and dirty" alternative to traditional qualitative data analysis.

Example Data

- As we've been using a few times this semester, `theoffice` dataset from the `schrute` package contains text data, which we can mine!

```
library(tidyverse)
library(schrute)
data('theoffice')
theoffice |>
  select(text) |>
  head(5)
```

```
# A tibble: 5 x 1
  text
  <chr>
1 All right Jim. Your quarterlies look very good. How are things at the library
2 Oh, I told you. I couldn't close it. So...
3 So you've come to the master for guidance? Is this what you're saying, grassh
4 Actually, you called me in here, but yeah.
5 All right. Well, let me show you how it's done.
```

Organizing the Data

- We discussed earlier in the semester the concept of "tidy" data where:
  1. Each variable is a column
  2. Each row is an observation
  3. Each cell represents or contains a value

- The way the data are organized at present doesn't allow for the traditional type of analysis, so we need to tidy it up in such a way that does.

- For us, this means *tokenizing* the text.

Organizing the Data

- Through tokenization, what we are doing is considering each word (could be more than one word but traditionally one word) as its own row.
- Consider the below string of text:

```
text_string <- tibble(
  text="Identity theft is not a joke, Jim!"
  )
```

Organizing the Data

- We can break this sentence down into individual rows by using the `unnest_tokens` function within the `tidytext` package:

```
library(tidytext)
text_string |>
  unnest_tokens(word,text)
```

```
# A tibble: 7 x 1
  word
  <chr>
1 identity
2 theft
3 is
4 not
5 a
6 joke
7 jim
```

Organizing the Data

- Let's tokenize the season 7 dialogue:

```r
s7 <- theoffice |>
  filter(season == 7) |>
  select(text) |>
  unnest_tokens(word,text)

s7 |>
  head()
```

```
# A tibble: 6 x 1
  word
  <chr>
1 you
2 fallin
3 behind
4 wuphf.com
5 ryan
6 we're
```

Organizing the Data

- Okay cool! But we have another problem.
- As you may notice, while we have the data in a nice tidy format, we have a lot of information that isn't really relevant.
- For example, it probably isn't pertinent for us to know how many times the words "the," "or," "we," or "and" are used.
  - These are called *stop words*. We typically want to remove stop words from our data prior to analysis.
- We can do this using a dataset called `stop_words` which is part of the `tidytext` package.
  - However, there actually exist several stop word datasets, many of which are contained in the `stopwords` package in R. We can join these separate datasets to make a larger stop words collection:

Organizing the Data

```r
library(stopwords)

data("stop_words")
data("data_stopwords_smart")
data("data_stopwords_snowball")
data("data_stopwords_stopwordsiso")

## Join all into one big dataset ##

stoppr <- tibble(word = c(data_stopwords_smart$en,
                          data_stopwords_snowball$en,
                          data_stopwords_stopwordsiso$en)
) |>
  distinct()
```

Organizing the Data

- Notice below, when we perform the anti join to remove the stop words, some of the words that were present at the beginning of the last dataset have been removed here:

```
s7_stop <- s7 |>
  anti_join(stoppr)

s7_stop |>
  head()
```

```
# A tibble: 6 x 1
  word
  <chr>
1 fallin
2 wuphf.com
3 ryan
4 dance
5 build
6 business
```

Organizing the Data

- Another issue we sometimes encounter with text data is when words with the same root exist.
  - For example, "fishing," "fished," and "fisher" all have the same root of "fish".
- The process of reducing these similar variations of words to their common root is called *word stemming*.
- We can use the `wordStem` function within the `SnowballC` package to do this for us:

Organizing the Data

- Notice that word stemming doesn't always do a perfect job which is why I created a separate column for the stemmed words and the unstemmed words.

```
library(SnowballC)
s7_stop <- s7_stop |>
  mutate(word2 = wordStem(word))

s7_stop |>
  head()
```

```
# A tibble: 6 x 2
  word      word2
  <chr>     <chr>
1 fallin    fallin
2 wuphf.com wuphf.com
3 ryan      ryan
4 dance     danc
5 build     build
6 business  busi
```

Analyzing the Data: Frequencies

- A very straightforward question we might have about our data is word frequencies! We already have learned how to use `dplyr` to figure this out:

```
s7_stop |>
  count(word,sort=T) |>
  head()
```

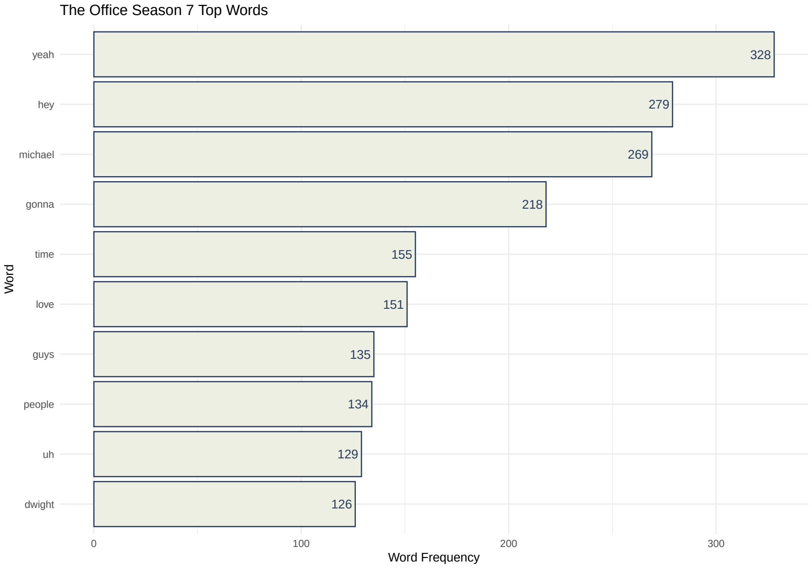
```
# A tibble: 6 x 2
  word        n
  <chr>   <int>
1 yeah      328
2 hey       279
3 michael   269
4 gonna     218
5 time      155
6 love      151
```

Analyzing the Data: Frequencies

- We have already learned how to generate horizontal, ordered bar charts, so let's do that for the top 10 words!

```
s7_stop |>
  count(word,sort=T) |>
  head(10) |>
  ggplot(aes(x=n,y=reorder(word,n))) +
  geom_bar(stat='identity',fill='#EDEFE2',
           color='#2A3C5F') +
  geom_text(aes(label=n),hjust=1.15,
            color="#2A3C5F") +
  labs(x="Word Frequency",
       y="Word",
       title="The Office Season 7 Top Words") +
  theme_minimal()
```

Analyzing the Data: Frequencies



Analyzing the Data: Frequencies

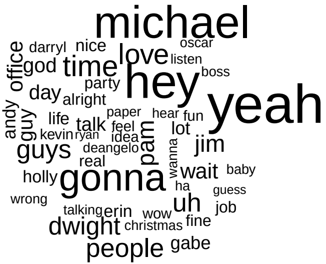
- We can also create word clouds to visualize word frequencies using the `wordcloud` package!

```
library(wordcloud)

s7_frequencies <- s7_stop |>
  count(word)

wordcloud(words = s7_frequencies$word,
          freq = s7_frequencies$n,
          max.words = 50)
```

Analyzing the Data: Frequencies



Analyzing the Data: Sentiment Analysis

- Beyond looking at word frequencies, we may also be interested in the intent or sentiment behind the words.
- For example, brands may want to follow social media posts about their company to determine whether what is trending online is positive or negative.
  - Sony's inability to differentiate between positive and negative online sentiment cost them a good deal with their movie Morbius: https://screenrant.com/morbius-box-office-flop-theaters-details/
- Like with stop words, we have to have dictionaries which assign a score or classification to specific words.
  - The `tidytext` package contains 3 sentiment dictionaries we can use.

Analyzing the Data: Sentiment Analysis

- Let's use the Bing sentiment dictionary to determine the top positive and negative words!

```
sentiment <- s7_stop |>
  select(word) |>
  inner_join(get_sentiments('bing'))

sentiment |>
  head()
```

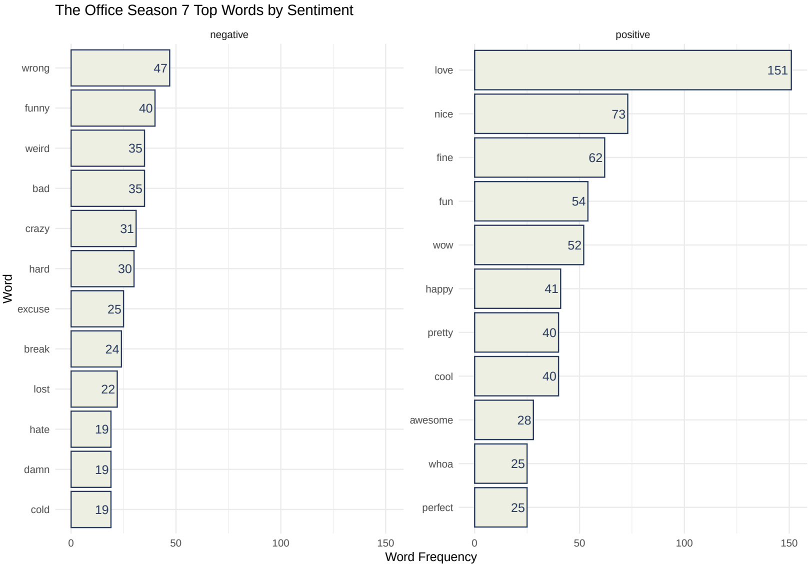
```
# A tibble: 6 x 2
  word  sentiment
  <chr> <chr>
1 hard  negative
2 love  positive
3 crazy negative
4 sad   negative
5 virus negative
6 lost  negative
```

Analyzing the Data: Sentiment Analysis

- We can generate horizontal, ordered bar charts looking at the top positive and negative words!

```
sentiment |>
  group_by(sentiment) |>
  count(word,sentiment,sort=T) |>
  ungroup() |>
  group_by(sentiment) |>
  slice_max(n,n=10) |>
  ggplot(aes(x=n,y=reorder(word,n))) +
  geom_bar(stat='identity',fill='#EDEFE2',
           color='#2A3C5F') +
  geom_text(aes(label=n),hjust=1.15,
            color="#2A3C5F") +
  facet_wrap(~sentiment,scales='free_y') +
  labs(x="Word Frequency",
       y="Word",
       title="The Office Season 7 Top Words by Sentiment") +
  theme_minimal()
```

Analyzing the Data: Sentiment Analysis



Analyzing the Data: Sentiment Analysis

- We can do something similar with word clouds!

```
library(reshape2)
sentiment |>
  count(word, sentiment, sort = TRUE) |>
  acast(word ~ sentiment, value.var = "n", fill = 0) |>
  comparison.cloud(colors = c("gray20", "pink"),
                   max.words = 100,
                   title.size=2)
```

Analyzing the Data: Sentiment Analysis