Lab 08 - Text Mining

```
knitr::opts_chunk$set(eval = F, include  = T)
```

Learning goals

- Use `unnest_tokens()` and `unnest_ngrams()` to extract tokens and ngrams from text.
- Use dplyr and ggplot2 to analyze text data

Lab description

For this lab we will be working with the medical record transcriptions from https://www.mtsamples.com/. And is loaded and "fairly" cleaned at https://github.com/JSC370/jsc370-2022/blob/main/data/medical_transcriptions/mtsamples.csv.

This markdown document should be rendered using `github_document` document.

Setup packages

You should load in `dplyr`, (or `data.table` if you want to work that way), `ggplot2` and `tidytext`. If you don't already have `tidytext` then you can install with

```
install.packages("tidytext")
```

read in Medical Transcriptions

Loading in reference transcription samples from https://www.mtsamples.com/

```
library(tidytext)
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(wordcloud)
library(kableExtra)

mt_samples <- read_csv("https://raw.githubusercontent.com/JSC370/jsc370-2022/main/data/medical_transcrip
mt_samples <- mt_samples %>%
  select(description, medical_specialty, transcription)

head(mt_samples)
```

Question 1: What specialties do we have?

We can use `count()` from `dplyr` to figure out how many different catagories do we have? Are these catagories related? overlapping? evenly distributed?

```
mt_samples %>%
  count(medical_specialty, sort = TRUE) %>%
  ggplot(aes(medical_specialty, n)) +
  geom_col() +
  coord_flip()
```

By looking at the above barplot, we can see there are 40 total categories in the dataset, and while some (such as surgery) are more frequent than others, there seems to be a somewhat even distribution of categories.

Question 2

- Tokenize the the words in the `transcription` column
- Count the number of times each token appears
- Visualize the top 20 most frequent words

```
tokens <- mt_samples %>%
  select(transcription) %>%
  unnest_tokens(word, transcription) %>%
  group_by(word) %>%
  summarise(word_frequency = n()) %>%
  arrange(across(word_frequency, desc)) %>%
  head(20)

tokens %>%
  kable() %>%
  kable_styling()
```

Above, we can see the top 20 words and their associated frequencies. Next, we will draw a barplot describing these frequencies:

```
tokens %>%
  ggplot(aes(reorder(word, -word_frequency), word_frequency)) +
  geom_bar(stat = 'identity')
```

We can also draw a (currently not so interesting) Wordcloud visualization of these words:

```
wordcloud(tokens$word, tokens$word_frequency)
```

Explain what we see from this result. Does it makes sense? What insights (if any) do we get?

The results we get from tokenizing make sense as words such as the, and, was, and so on are typically assumed to be frequently occurring in English medical journals. However, we don't gain much insight from this as knowing these regular English words are the most prevalent doesn't tell us much about the contents of these journals, the categories to which they belong, or frankly any other useful information.

Question 3

- Redo visualization but remove stopwords before
- Bonus points if you remove numbers as well

```
tokens <- mt_samples %>%
  select(transcription) %>%
  unnest_tokens(word, transcription) %>%
  anti_join(stop_words, by="word") %>%
  subset(!grepl("^\\d+$", word)) %>%
  group_by(word) %>%
  summarise(word_frequency = n()) %>%
  arrange(across(word_frequency, desc)) %>%
  head(20)

tokens %>%
  kable() %>%
  kable_styling()
```

Above, we can see the top 20 words and their associated frequencies AFTER filtering out stopwords. Next, we will draw a barplot describing these frequencies:

```
tokens %>%
  ggplot(aes(reorder(word, -word_frequency), word_frequency)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```

We can also draw a (now a little more interesting) Wordcloud visualization of these words:

```
wordcloud(tokens$word, tokens$word_frequency)
```

What do we see know that we have removed stop words? Does it give us a better idea of what the text is about?

By removing stopwords we can indeed start to see some more interesting results about what words appear the most frequently. That is, it is now a lot clearer that we are looking through medical journals with words such as “patient”, “procedure”, and “diagnosis” appearing very frequently.

Question 4

Repeat question 2, but this time tokenize into bi-grams. How does the result change if you look at tri-grams?

We will first consider bigrams:

```
tokens <- mt_samples %>%
  select(transcription) %>%
  unnest_tokens(bigram, transcription, token = "ngrams", n = 2) %>%
  group_by(bigram) %>%
  summarise(bigram_frequency = n()) %>%
```

```
  separate(bigram,
           c("word1", "word2"),
           extra = "drop",
           remove = FALSE,
           sep = " ",
           fill = "right") %>%
  anti_join(stop_words, by = c("word1" = "word")) %>%
  anti_join(stop_words, by = c("word2" = "word")) %>%
  subset(!grepl("\\d+", bigram)) %>%
  arrange(across(bigram_frequency, desc)) %>%
  head(20)

tokens %>%
  kable() %>%
  kable_styling()
```

Above, we can see the top 20 bigrams and their associated frequencies AFTER filtering out stopwords. Next, we will draw a barplot describing these frequencies:

```
tokens %>%
  ggplot(aes(reorder(bigram, -bigram_frequency), bigram_frequency)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```

Now, we will do the exact same thing but with trigrams:

```
tokens <- mt_samples %>%
  select(transcription) %>%
  unnest_tokens(trigram, transcription, token = "ngrams", n = 3) %>%
  group_by(trigram) %>%
  summarise(trigram_frequency = n()) %>%
  separate(trigram,
           c("word1", "word2", "word3"),
           extra = "drop",
           remove = FALSE,
           sep = " ",
           fill = "right") %>%
  anti_join(stop_words, by = c("word1" = "word")) %>%
  anti_join(stop_words, by = c("word2" = "word")) %>%
  anti_join(stop_words, by = c("word3" = "word")) %>%
  subset(!grepl("\\d+", trigram)) %>%
  arrange(across(trigram_frequency, desc)) %>%
  head(20)

tokens %>%
  kable() %>%
  kable_styling()
```

Above, we can see the top 20 trigrams and their associated frequencies AFTER filtering out stopwords. Next, we will draw a barplot describing these frequencies:

```
tokens %>%
  ggplot(aes(reorder(trigram, -trigram_frequency), trigram_frequency)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```

We can see that the results change somewhat after transitioning from bigrams to trigrams, which makes sense because chunks of 3 words will probably have different frequencies than chunks of 2 words. This could be for many reasons, such as some of the more frequent bigrams having a stop word immediately following (i.e. "blood loss of"), which would make the corresponding trigram ineligible to be included in our count.

Question 5

Using the results you got from question 4. Pick a word and count the words that appears after and before it.

First, reset `tokens` to hold all the trigrams and not just the top 20:

```
tokens <- mt_samples %>%
  select(transcription) %>%
  unnest_tokens(trigram, transcription, token = "ngrams", n = 3) %>%
  group_by(trigram) %>%
  summarise(trigram_frequency = n()) %>%
  separate(trigram,
           c("word1", "word2", "word3"),
           extra = "drop",
           remove = FALSE,
           sep = " ",
           fill = "right") %>%
  anti_join(stop_words, by = c("word1" = "word")) %>%
  anti_join(stop_words, by = c("word2" = "word")) %>%
  anti_join(stop_words, by = c("word3" = "word")) %>%
  subset(!grepl("\\d+", trigram)) %>%
  arrange(across(trigram_frequency, desc))
```

Then, we will look at which words appear the most frequently before the word "blood":

```
tokens %>%
  subset(word2 == "blood") %>%
  group_by(word1) %>%
  summarise(word1_freq = n()) %>%
  arrange(across(word1_freq, desc)) %>%
  head(20) %>%
  kable() %>%
  kable_styling()
```

We can also visualize the frequencies:

```
tokens %>%
  subset(word2 == "blood") %>%
  group_by(word1) %>%
  summarise(word1_freq = n()) %>%
  arrange(across(word1_freq, desc)) %>%
  head(20) %>%
  ggplot(aes(word1, word1_freq)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```

We can also look at which words appear the most frequently AFTER the word "blood":

```
tokens %>%
  subset(word2 == "blood") %>%
  group_by(word3) %>%
  summarise(word3_freq = n()) %>%
  arrange(across(word3_freq, desc)) %>%
  head(20) %>%
  kable() %>%
  kable_styling()
```

We can again visualize the frequencies:

```
tokens %>%
  subset(word2 == "blood") %>%
  group_by(word3) %>%
  summarise(word3_freq = n()) %>%
  arrange(across(word3_freq, desc)) %>%
  head(20) %>%
  ggplot(aes(word3, word3_freq)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```

From the visualizations and frequency tables above, we can conclude that the word that most frequently appears before "blood" is "patient's", while the word that most frequently appears after "blood" is "pressure".

Question 6

Which words are most used in each of the specialties. you can use `group_by()` and `top_n()` from `dplyr` to have the calculations be done within each specialty. Remember to remove stopwords. How about the most 5 used words?

First, we will examine the top words used in each of the specialties:

```
mt_samples %>%
  unnest_tokens(word, transcription) %>%
  anti_join(stop_words, by = "word") %>%
  subset(!grepl("^\\d+$", word)) %>%
  group_by(medical_specialty) %>%
```

```
  count(word, sort = TRUE) %>%
  top_n(1, n) %>%
  kable() %>%
  kable_styling()
```

Above, we have a table describing which word is used most frequently in each specialty and how frequently the specialty occurs. Next, we can look at the top 5 most frequently used words for each specialty:

```
mt_samples %>%
  unnest_tokens(word, transcription) %>%
  anti_join(stop_words, by = "word") %>%
  subset(!grepl("^\\d+$", word)) %>%
  group_by(medical_specialty) %>%
  count(word) %>%
  top_n(5, n) %>%
  kable() %>%
  kable_styling()
```

Question 7 - extra

Find your own insight in the data:

Ideas:

- Interesting ngrams
- See if certain words are used more in some specialties then others

Deliverables

1. Questions 1-7 answered, pdf or html output uploaded to quercus