```r
# Project 2 - UseNet Sci. Data - Axl Ibiza, MBA
# 27 September 2024
# Code adapted from https://github.com/dgrtwo/tidy-text-mining/tree/master/data

# Install packages

# install.packages("tidytext")
# install.packages("widyr")
# install.packages("ggraph")
# install.packages("igraph")
# install.packages("topicmodels")
# install.packages("textdata")

# Load packages
library(dplyr)
library(ggraph)
library(ggplot2)
library(igraph)
library(purrr)
library(readr)
library(stringr)
library(textdata)
library(tidytext)
library(tidyr)
library(topicmodels)
library(widyr)

# Load the data
training_folder <- "data\20news-bydate\20news-bydate-train"

# Define a function to read all files from a folder into a data frame
read_folder <- function(infolder) {
  tibble(file = dir(infolder, full.names = TRUE)) %>%
    mutate(text = map(file, read_lines)) %>%
    transmute(id = basename(file), text) %>%
    unnest(text)
}

# Use unnest() and map() to apply read_folder to each subfolder
raw_text <- tibble(folder = dir(training_folder, full.names = TRUE)) %>%
  mutate(folder_out = map(folder, read_folder)) %>%
  unnest(cols = c(folder_out)) %>%
  transmute(newsgroup = basename(folder), c(id), c(text))

load("data/raw_text.rda")
raw_text

# Plot distinct messages per newsgroup
raw_text %>%
  group_by(newsgroup) %>%
  summarize(messages = n_distinct(id)) %>%
  ggplot(aes(messages, newsgroup)) +
  geom_col() +
  labs(y = NULL)

### Preprocess text

# must occur after the first occurrence of an empty line,
# and before the first occurrence of a line starting with --
cleaned_text <- raw_text %>%
  group_by(newsgroup, id) %>%
  filter(cumsum(text == "") > 0,
         cumsum(str_detect(text, "^--")) == 0) %>%
  ungroup()
```

```
# We also choose to manually remove two messages, `9704` and `9985` that contained a large
amount of non-text content.
cleaned_text <- cleaned_text %>%
  filter(str_detect(text, "^[^>]+[A-Za-z\\d]") | text == "",
         !str_detect(text, "writes(:|\\.\\.\\.)$"),
         !str_detect(text, "^In article <"),
         !id %in% c(9704, 9985))

# At this point, we're ready to use `unnest_tokens()` to split the dataset into tokens,
while removing stop-words.
usenet_words <- cleaned_text %>%
  unnest_tokens(word, text) %>%
  filter(str_detect(word, "[a-z']$"),
         !word %in% stop_words$word)

### Words in Newsgroups
usenet_words %>%
  count(word, sort = TRUE)

words_by_newsgroup <- usenet_words %>%
  count(newsgroup, word, sort = TRUE) %>%
  ungroup()
words_by_newsgroup

### tf-idf within newsgroups
tf_idf <- words_by_newsgroup %>%
  bind_tf_idf(word, newsgroup, n) %>%
  arrange(desc(tf_idf))
tf_idf

tf_idf %>%
  filter(str_detect(newsgroup, "^sci\\.")) %>%
  group_by(newsgroup) %>%
  slice_max(tf_idf, n = 12) %>%
  ungroup() %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(tf_idf, word, fill = newsgroup)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ newsgroup, scales = "free") +
  labs(x = "tf-idf", y = NULL)

newsgroup_cors <- words_by_newsgroup %>%
  pairwise_cor(newsgroup, word, n, sort = TRUE)
newsgroup_cors

set.seed(2017)

newsgroup_cors %>%
  filter(correlation > .4) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = correlation, width = correlation)) +
  geom_node_point(size = 6, color = "lightblue") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()

### Topic Modeling
# include only words that occur at least 50 times
word_sci_newsgroups <- usenet_words %>%
  filter(str_detect(newsgroup, "^sci")) %>%
  group_by(word) %>%
  mutate(word_total = n()) %>%
  ungroup() %>%
  filter(word_total > 50)
```

```
# convert into a document-term matrix
# with document names such as sci.crypt_14147
sci_dtm <- word_sci_newsgroups %>%
  unite(document, newsgroup, id) %>%
  count(document, word) %>%
  cast_dtm(document, word, n)

sci_lda <- LDA(sci_dtm, k = 4, control = list(seed = 2016))

sci_lda %>%
  tidy() %>%
  group_by(topic) %>%
  slice_max(beta, n = 8) %>%
  ungroup() %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()

sci_lda %>%
  tidy(matrix = "gamma") %>%
  separate(document, c("newsgroup", "id"), sep = "_") %>%
  mutate(newsgroup = reorder(newsgroup, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ newsgroup) +
  labs(x = "Topic",
       y = "# of messages where this was the highest % topic")

### Sentiment Analysis

# Load the AFINN sentiment lexicon
load("data/afinn.rda")

# Calculate newsgroup sentiments
newsgroup_sentiments <- words_by_newsgroup %>%
  inner_join(afinn, by = "word") %>%
  group_by(newsgroup) %>%
  filter(str_detect(newsgroup, "^sci")) %>%
  summarize(value = sum(value * n) / sum(n))

# Plot the newsgroup sentiments
newsgroup_sentiments %>%
  mutate(newsgroup = reorder(newsgroup, value)) %>%
  ggplot(aes(value, newsgroup, fill = value > 0)) +
  geom_col(show.legend = FALSE) +
  labs(x = "Average sentiment value", y = NULL)

# Calculate contributions
contributions <- usenet_words %>%
  inner_join(afinn, by = "word") %>%
  group_by(word) %>%
  summarize(occurences = n(),
            contribution = sum(value))

contributions %>%
  slice_max(abs(contribution), n = 25) %>%
  mutate(word = reorder(word, contribution)) %>%
  ggplot(aes(contribution, word, fill = contribution > 0)) +
  geom_col(show.legend = FALSE) +
  labs(y = NULL)

## Sentiment Analysis by Word
```

```r
top_sentiment_words <- words_by_newsgroup %>%
  inner_join(afinn, by = "word") %>%
  mutate(contribution = value * n / sum(n))
top_sentiment_words %>%
  filter(str_detect(newsgroup, "^sci")) %>%
  group_by(newsgroup) %>%
  slice_max(abs(contribution), n = 12) %>%
  ungroup() %>%
  mutate(newsgroup = reorder(newsgroup, contribution),
         word = reorder_within(word, contribution, newsgroup)) %>%
  ggplot(aes(contribution, word, fill = contribution > 0)) +
  geom_col(show.legend = FALSE) +
  scale_y_reordered() +
  facet_wrap(~ newsgroup, scales = "free") +
  labs(x = "Sentiment value * # of occurrences", y = NULL)

## Sentiment Analysis by Message
sentiment_messages <- usenet_words %>%
  inner_join(afinn, by = "word") %>%
  filter(str_detect(newsgroup, "^sci")) %>%
  group_by(newsgroup, id) %>%
  summarize(sentiment = mean(value),
            words = n()) %>%
  ungroup() %>%
  filter(words >= 5) %>%
  arrange(desc(sentiment))

# What were the most positive messages in sci?
sentiment_messages %>%
  arrange(desc(sentiment))
sentiment_messages

# Function to check text of specific message
print_message <- function(group, message_id) {
  result <- cleaned_text %>%
    filter(newsgroup == group, id == message_id, text != "")

  cat(result$text, sep = "\n")
}

print_message("sci.space", 61094)
print_message("sci.electronics", 53836)
print_message("sci.med", 58863)
print_message("sci.crypt", 15304)

### n-gram Analysis
usenet_bigrams <- cleaned_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

usenet_bigram_counts <- usenet_bigrams %>%
  count(newsgroup, bigram, sort = TRUE) %>%
  separate(bigram, c("word1", "word2"), sep = " ")

negate_words <- c("not", "without", "no", "can't", "don't", "won't")

usenet_bigram_counts %>%
  filter(word1 %in% negate_words) %>%
  count(word1, word2, wt = n, sort = TRUE) %>%
  inner_join(afinn, by = c(word2 = "word")) %>%
  mutate(contribution = value * n) %>%
  group_by(word1) %>%
  slice_max(abs(contribution), n = 10) %>%
  ungroup() %>%
  mutate(word2 = reorder_within(word2, contribution, word1)) %>%
  ggplot(aes(contribution, word2, fill = contribution > 0)) +
geom_col(show.legend = FALSE) +
facet_wrap(~ word1, scales = "free", nrow = 3) +
scale_y_reordered() +
labs(x = "Sentiment value * # of occurrences",
     y = "Words preceded by a negation")
```