CSCI_6410_Practical_Assignment_2

Shubham Bamane

2024-06-02

```
knitr::opts_chunk$set(echo = TRUE)
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Practical 2: Electronic Medical Records

```
library(tidyr)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(clinspacy)
```

```
## Warning: package 'clinspacy' was built under R version 4.3.3
```

```
## Welcome to clinspacy.
```

```
## By default, this package will install and use miniconda and create a "clinspacy" conda environment.
```

```
## If you want to override this behavior, use clinspacy_init(miniconda = FALSE) and specify an alternat
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.3.3
```

```
library(stringr)
```

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 4.3.3
```

```
library('reshape2')
```

```
## Warning: package 'reshape2' was built under R version 4.3.3
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
raw.data <- clinspacy::dataset_mtsamples()
dplyr::glimpse(raw.data)
```

```
## Rows: 4,999
## Columns: 6
## $ note_id           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ description       <chr> "A 23-year-old white female presents with complaint ~
## $ medical_specialty <chr> "Allergy / Immunology", "Bariatrics", "Bariatrics", ~
## $ sample_name       <chr> "Allergic Rhinitis", "Laparoscopic Gastric Bypass Co~
## $ transcription     <chr> "SUBJECTIVE:,  This 23-year-old white female present~
## $ keywords          <chr> "allergy / immunology, allergic rhinitis, allergies,~
```

1. Using the output of dplyr's glimpse command (or rstudio's data viewer by clicking on raw.data in the Environment pane) provide a description of what you think each variable in this dataset contains. Based on the glimpse output, the following is probably what each column contains:

note_id: This column, which is an integer, most likely contains a unique identifier for every medical note. This column aids in uniquely identifying each row in the dataset, which corresponds to a different medical note.

description: The medical case or patient presentation is briefly described in this character (chr) type column. As an instance, "A 23-year-old white female presents with complaint of. . . "

medical_specialty: The medical specialty associated with the note is indicated in this character column. It might say "Allergy / Immunology," "Bariatrics," etc. as an example. This makes it easier to classify the notes according to the medical specialty they cover.

sample_name: The name of the medical case or sample, such as "Laparoscopic Gastric Bypass Complications," "Allergic Rhinitis," etc., appears in this character column. It gives the case mentioned in the note a succinct title.

transcription: The entire text of the medical transcription is contained in this additional character column. It includes narrative content that has been meticulously transcribed from the medical notes.

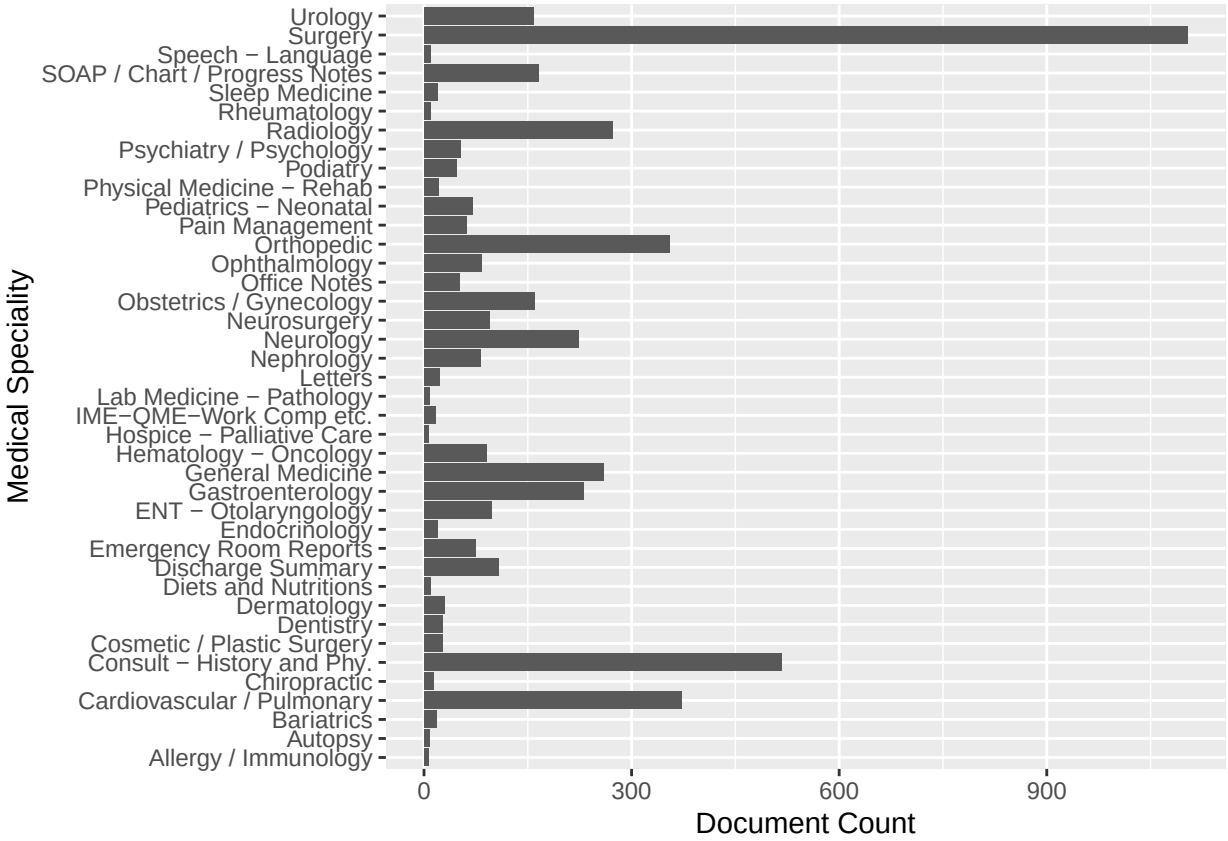
keywords: A list of keywords related to the medical note is provided in this character column. These keywords, which include phrases like "allergy / immunology, allergic rhinitis, allergies," etc., can be helpful for searches and classification.

Let's see how many different medical specialties are featured in these notes:

```
raw.data %>% dplyr::select(medical_specialty) %>% dplyr::n_distinct()
```

```
## [1] 40
```

```
ggplot2::ggplot(raw.data, ggplot2::aes(y=medical_specialty)) + ggplot2::geom_bar() + labs(x="Document C
```




```
filtered.data <- raw.data %>% dplyr::filter(medical_specialty %in% c("Orthopedic", "Radiology", "Surgery
```

Text Processing

```
analysis.data <- filtered.data %>%
  unnest_tokens(word, transcription) %>%
  mutate(word = str_replace_all(word, "[^[:alnum:]]", "")) %>%
  filter(!str_detect(word, "[0-9]")) %>%
  anti_join(stop_words) %>%
  group_by(note_id) %>%
  summarise(transcription = paste(word, collapse = " ")) %>%
  left_join(select(filtered.data, -transcription), by = "note_id")
```

```
## Joining with `by = join_by(word)`
```

```
tokenized.data.unigram <- analysis.data %>% tidytext::unnest_tokens(word, transcription, to_lower=TRUE)
```

```
tokenized.data <- analysis.data %>% tidytext::unnest_tokens(ngram, transcription, token = "ngrams", n=2
```

```
tidytext::stop_words %>% dplyr::group_by(lexicon) %>% dplyr::distinct(word) %>% dplyr::summarise(n=dplyr
```

```
## # A tibble: 3 x 2
##   lexicon      n
##   <chr>    <int>
## 1 SMART      570
## 2 onix       398
## 3 snowball   174
```

```
filtered.data <- raw.data %>%
  filter(medical_specialty %in% c("Orthopedic", "Radiology", "Surgery"))
```

```
# We clean the transcription text while removing the stop words.

analysis.data <- filtered.data %>%
  unnest_tokens(word, transcription) %>%
  mutate(word = str_replace_all(word, "[^[:alnum:]]", "")) %>%
  filter(!str_detect(word, "[0-9]")) %>%
  anti_join(tidytext::stop_words, by = "word") %>%
  group_by(note_id) %>%
  summarise(transcription = paste(word, collapse = " ")) %>%
  left_join(select(filtered.data, -transcription), by = "note_id")
```

```
# To count unique unigrams per specialty

unique_unigrams <- analysis.data %>%
  unnest_tokens(word, transcription, to_lower = TRUE) %>%
  group_by(medical_specialty) %>%
  summarise(unique_unigrams = n_distinct(word))
```

```
print(unique_unigrams)
```

2. How many unique unigrams are there in the transcripts from each specialty

```
## # A tibble: 3 x 2
##   medical_specialty unique_unigrams
##   <chr>                       <int>
## 1 Orthopedic                   7681
## 2 Radiology                    5933
## 3 Surgery                     11977
```

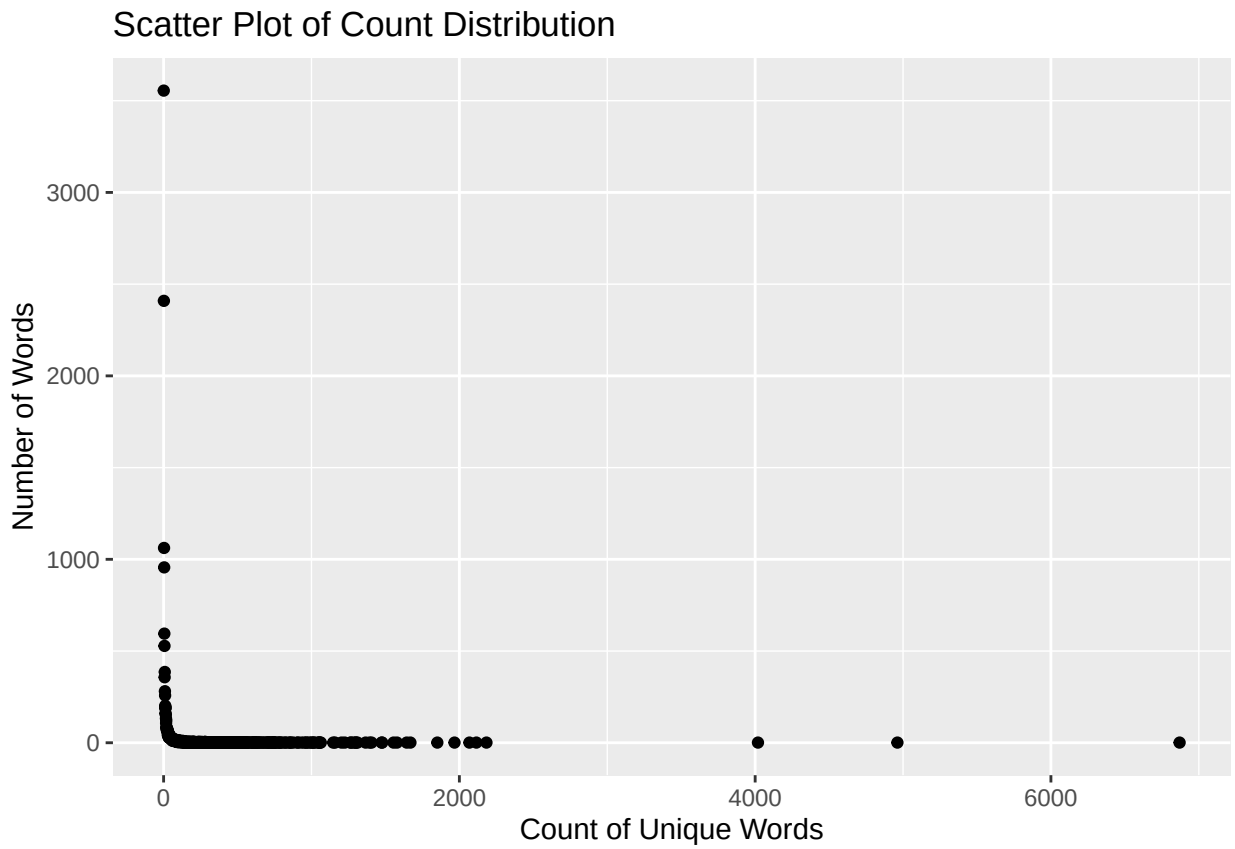
Based on the analysis, here are the counts of unique unigrams in the transcripts from each specialty:

Orthopedic: 7,681 unique unigrams Radiology: 5,933 unique unigrams Surgery: 11,977 unique unigrams These counts represent the number of distinct words (unigrams) found in the medical transcripts for each of the selected specialties.

Let's plot some distribution of unigram tokens (words)

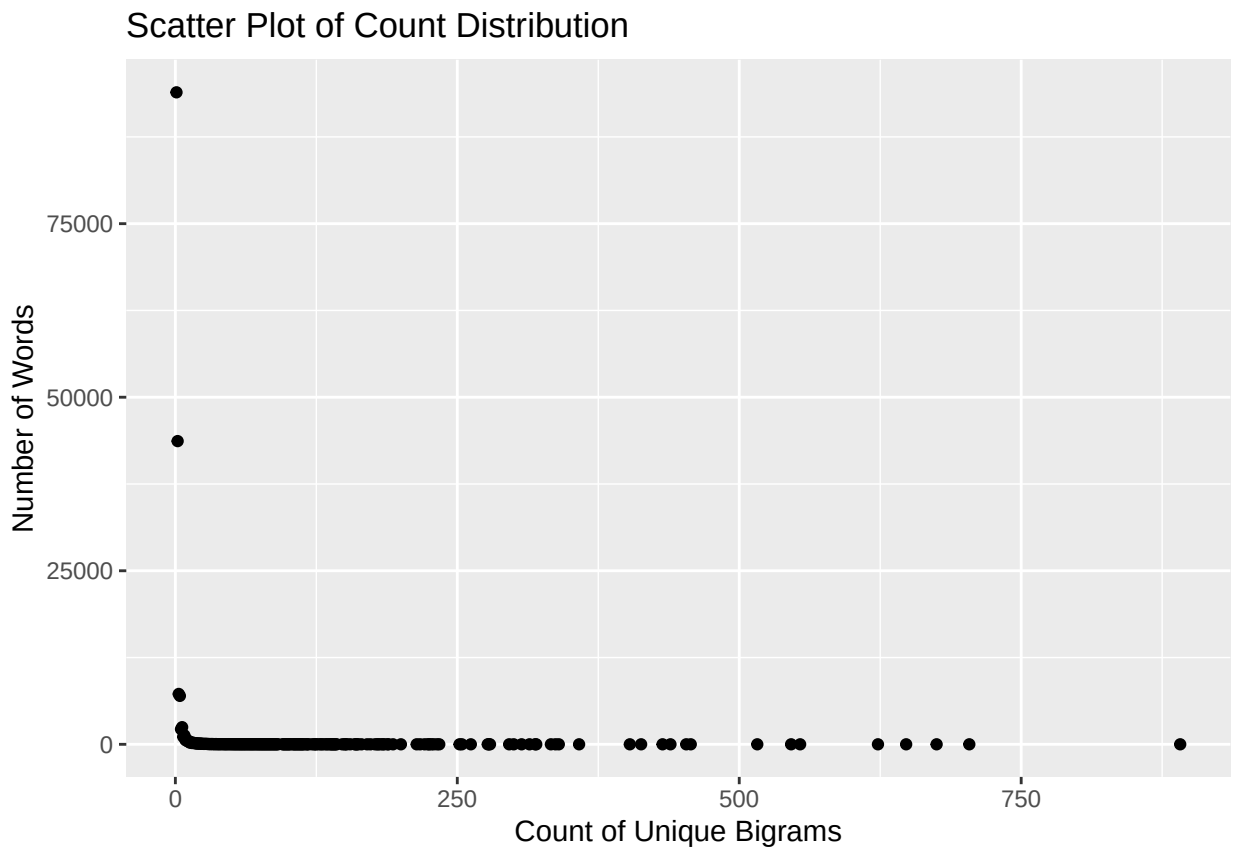
```
word_counts <- tokenized.data.unigram %>%
    group_by(word) %>%
    summarise(count = n()) %>%
    ungroup() %>%
    arrange(desc(count))

count_distribution <- word_counts %>%
  group_by(count) %>%
  summarise(num_words = n()) %>%
  ungroup()

 ggplot2::ggplot(count_distribution, aes(x = count, y = num_words)) +
  geom_point() +
```

```
  labs(title = "Scatter Plot of Count Distribution",
       x = "Count of Unique Words",
       y = "Number of Words")
```



Let's plot some distribution of bigram tokens (words)

```
word_counts <- tokenized.data %>%
    group_by(ngram) %>%
    summarise(count = n()) %>%
    ungroup() %>%
    arrange(desc(count))

count_distribution <- word_counts %>%
  group_by(count) %>%
  summarise(num_words = n()) %>%
  ungroup()

 ggplot2::ggplot(count_distribution, aes(x = count, y = num_words)) +
  geom_point() +
  labs(title = "Scatter Plot of Count Distribution",
       x = "Count of Unique Bigrams",
       y = "Number of Words")
```




```
# Filter data (same specialties)
filtered.data <- raw.data %>%
  dplyr::filter(medical_specialty %in% c("Orthopedic", "Radiology", "Surgery"))

# Text pre-processing (same steps)
analysis.data <- filtered.data %>%
  unnest_tokens(word, transcription) %>%
  mutate(word = str_replace_all(word, "[^[:alnum:]]", "")) %>%
  filter(!str_detect(word, "[0-9]")) %>%
  anti_join(stop_words) %>%
  group_by(note_id) %>%
  summarise(transcription = paste(word, collapse = " ")) %>%
  left_join(select(filtered.data, -transcription), by = "note_id")
```

3. How many unique bi-grams are there in each category without stop words and numbers?

```
## Joining with `by = join_by(word)`
```

```
# Count unique bigrams in each specialty
bigram.counts <- analysis.data %>%
  unnest_tokens(bigram, transcription, token = "ngrams", n = 2) %>%
  group_by(medical_specialty) %>%
  summarise(n_unique_bigrams = dplyr::n_distinct(bigram))

# Print results
print(bigram.counts)
```

```
## # A tibble: 3 x 2
##   medical_specialty n_unique_bigrams
##   <chr>                        <int>
## 1 Orthopedic                   55730
## 2 Radiology                    28294
## 3 Surgery                     130404
```

```
# Filter the data for the selected specialties
filtered.data <- raw.data %>%
  filter(medical_specialty %in% c("Orthopedic", "Radiology", "Surgery"))

# Tokenize to sentences
tokenized.data.sentence <- filtered.data %>%
  unnest_tokens(sentence, transcription, token = "sentences", to_lower = TRUE)

# Count unique sentences per specialty
unique_sentences <- tokenized.data.sentence %>%
  group_by(medical_specialty) %>%
  summarise(unique_sentences = n_distinct(sentence))

# Display the counts
print(unique_sentences)
```

4. How many unique sentences are there in each category?

```
## # A tibble: 3 x 2
##   medical_specialty unique_sentences
##   <chr>                        <int>
## 1 Orthopedic                   11174
## 2 Radiology                     4565
## 3 Surgery                      29652
```

Now that we've tokenized to words and removed stop words, we can find the most commonly word used within each category:

```
tokenized.data %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::count(ngram, sort = TRUE) %>%
  dplyr::top_n(5)
```

```
## Selecting by n
```

```
## # A tibble: 16 x 3
## # Groups:   medical_specialty [3]
##    medical_specialty ngram                       n
##    <chr>             <chr>                   <int>
##  1 Surgery           prepped draped            696
##  2 Surgery           preoperative diagnosis    555
##  3 Surgery           procedure patient         551
##  4 Surgery           postoperative diagnosis   518
##  5 Surgery           tolerated procedure       515
##  6 Orthopedic        prepped draped            183
##  7 Orthopedic        preoperative diagnosis    141
##  8 Orthopedic        lower extremity           139
##  9 Orthopedic        range motion              139
```

```
## 10 Orthopedic        postoperative diagnosis   124
## 11 Radiology         carotid artery             59
## 12 Radiology         heart rate                 52
## 13 Radiology         reason exam                51
## 14 Radiology         left ventricular           50
## 15 Radiology         coronary artery            43
## 16 Radiology         exam unremarkable          43
```

5. Do you think a general purpose lemmatizer will work well for medical data? Why might it not? Medical data may not be the best fit for a general-purpose lemmatizer for a number of reasons.

1. Terminology Specific to a Domain:

Specialised terms and abbreviations that may not be recognised by a general-purpose lemmatizer are frequently found in medical data. For instance, "mir" (missing the medical context) could be lemmatized incorrectly from "MRI" (Magnetic Resonance Imaging).

2. Word-Count Units:

Multiple words can be combined to form medical terms, which a general lemmatizer may not handle correctly. "White blood cell," for example, could be lemmatized as "white cell" (incorrectly separating the unit) or "white blood cell" (correct).

3. Abbreviations and Acronyms:

A general lemmatizer might not be able to expand or handle acronyms and abbreviations that are commonly used in medical data appropriately. Certain abbreviations, like "Rx" (prescription), may not have a clear lemma; therefore, expanding "Hgb" (haemoglobin) to "haemoglobin" may be necessary.

4. Complexity of Morphology:

A general lemmatizer may not be able to fully capture the complex morphology (word structure) of medical terms. For instance, "dermatologist" might be mistakenly lemmatized to "derma" due to the suffix "-tologist" being omitted.

Unfortunately, a specialised lemmatizer like in clinspacy is going to be very painful to install so we will just use a simple lemmatizer for now:

```
lemmatized.data <- tokenized.data %>% dplyr::mutate(lemma=textstem::lemmatize_words(ngram))
```

We can now calculate the frequency of lemmas within each specialty and note.

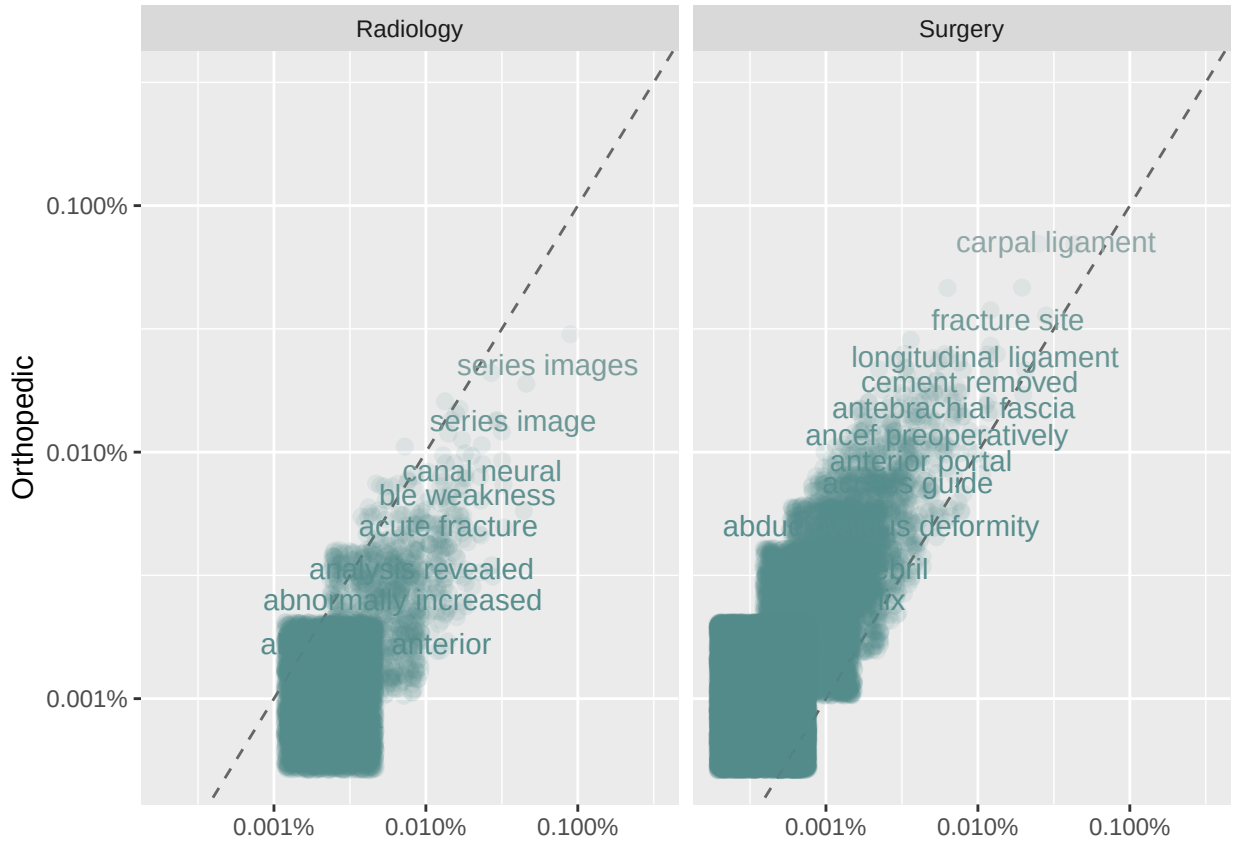
```
lemma.freq <- lemmatized.data %>%
  dplyr::count(medical_specialty, lemma) %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::mutate(proportion = n / sum(n)) %>%
  tidyr::pivot_wider(names_from = medical_specialty, values_from = proportion) %>%
  tidyr::pivot_longer(`Surgery`:`Radiology`,
               names_to = "medical_specialty", values_to = "proportion")
```

```
ggplot2::ggplot(lemma.freq, ggplot2::aes(x=proportion,
                                         y=`Orthopedic`,
                                         color=abs(`Orthopedic` - proportion))) +
  ggplot2::geom_abline(color="gray40", lty=2) +
  ggplot2::geom_jitter(alpha=0.1, size=2.5, width=0.3, height=0.3) +
  ggplot2::geom_text(ggplot2::aes(label=lemma), check_overlap=TRUE, vjust=1.5) +
  ggplot2::scale_x_log10(labels=scales::percent_format()) +
  ggplot2::scale_y_log10(labels=scales::percent_format()) +
  ggplot2::scale_color_gradient(limits=c(0, 0.001), low="darkslategray4", high="gray75") +
  ggplot2::facet_wrap(~medical_specialty, ncol = 2) +
```

```
  ggplot2::theme(legend.position="none") +
  ggplot2:: labs(y="Orthopedic", x = NULL)
```

```
## Warning: Removed 314400 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 314400 rows containing missing values or values outside the scale range
## (`geom_text()`).
```



6. What does this plot tell you about the relative similarity of lemma frequencies between Surgery and Orthopedic and between radiology and Surgery? Based on what these specialties involve, is this what you would expect?

A conclusive analysis is hindered by the missing data, but the plot reveals an intriguing pattern in the lemma (word root) frequencies among these medical specialties.

The "Orthopaedic" data points tend to cluster more closely with "Surgery" than with "Radiology." This suggests that orthopaedics and surgery have a greater degree of word similarity. This is consistent with how we understand these fields: a vocabulary pertaining to techniques, tools, drugs, and anatomical structures is probably shared by both orthopaedics (musculoskeletal procedures) and surgery (general surgical procedures).

Conversely, the data points pertaining to "Radiology" typically indicate a higher separation from both "Surgery" and "Orthopaedics." This suggests that the vocabularies of Radiology and the other two specialties are less similar. While surgery involves procedures, radiology concentrates on using medical imaging for diagnosis. Radiology probably uses a more distinct vocabulary when it comes to imaging techniques, anatomy visualisation, and interpretation, even though there may be some overlap in terminology related to diagnoses and imaging-guided procedures.

As a result, given the nature of these specialties, the plot makes sense. But it's important to recognise your

limitations. The absence of potentially important information is indicated by the missing data warnings, which affects the distribution of lemmas. Furthermore, "Orthopaedic" is the only one that the plot compares to the others. Comparing the three specialties would be a more thorough analysis.

Finally, the plot suggests that lemma frequencies and medical specialties may be related in a way that is instructive. But for a more thorough analysis, filling in the gaps in the data and looking into how all three specialties compare are essential.

```
ggplot2::ggplot(lemma.freq, ggplot2::aes(x = proportion,
                                         y = `Orthopedic`,
                                         color = abs(`Orthopedic` - proportion))) +
  ggplot2::geom_abline(color = "gray40", lty = 2) +
  ggplot2::geom_jitter(alpha = 0.1, size = 2.5, width=0.3, height=0.3) +
  ggplot2::geom_text(ggplot2::aes(label = lemma), check_overlap = TRUE, vjust = 1.5) +
  ggplot2::scale_x_log10(labels = scales::percent_format()) +
  ggplot2::scale_y_log10(labels = scales::percent_format()) +
  ggplot2::scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
  ggplot2::facet_wrap(~medical_specialty, ncol = 2, scales = "free") +
  ggplot2::theme(legend.position = "none") +
  ggplot2::labs(y = "Surgery or Radiology", x = NULL)
```

7. Modify the above plotting code to do a direct comparison of Surgery and Radiology (i.e., have Surgery or Radiology on the Y-axis and the other 2 specialties as the X facets)

```
## Warning: Removed 314400 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 314400 rows containing missing values or values outside the scale range
## (`geom_text()`).
```

TF-IDF Normalisation

Maybe looking at lemmas across all notes in a specialty is misleading, what if we look at lemma frequencies across a specialty.

```
lemma.counts <- lemmatized.data %>% dplyr::count(medical_specialty, lemma)
total.counts <- lemma.counts %>%
                      dplyr::group_by(medical_specialty) %>%
                      dplyr::summarise(total=sum(n))

all.counts <- dplyr::left_join(lemma.counts, total.counts)
```

```
## Joining with `by = join_by(medical_specialty)`
```

```
all.counts.tfidf <- tidytext::bind_tf_idf(all.counts, lemma, medical_specialty, n)
```

```
all.counts.tfidf %>% dplyr::group_by(medical_specialty) %>% dplyr::slice_max(order_by=tf_idf, n=10)
```

```
## # A tibble: 30 x 7
## # Groups:   medical_specialty [3]
##    medical_specialty lemma                   n total       tf   idf   tf_idf
##    <chr>             <chr>               <int> <int>    <dbl> <dbl>    <dbl>
##  1 Orthopedic        range motion          139 97774 0.00142  0.405 0.000576
##  2 Orthopedic        carpal ligament        85 97774 0.000869 0.405 0.000352
##  3 Orthopedic        transverse carpal      81 97774 0.000828 0.405 0.000336
##  4 Orthopedic        extremity prepped      79 97774 0.000808 0.405 0.000328
##  5 Orthopedic        proximal phalanx       75 97774 0.000767 0.405 0.000311
##  6 Orthopedic        department anesthesia  63 97774 0.000644 0.405 0.000261
```

```
##  7 Orthopedic        dissection carried       63 97774 0.000644 0.405 0.000261
##  8 Orthopedic        steri strips             59 97774 0.000603 0.405 0.000245
##  9 Orthopedic        closed vicryl            58 97774 0.000593 0.405 0.000241
## 10 Orthopedic        dressing applied         58 97774 0.000593 0.405 0.000241
## # i 20 more rows
```

8. Are there any lemmas that stand out in these lists? Why or why not? Yes, these lists do contain some particularly noteworthy lemmas. Within the orthopaedic specialty, terms like "range motion," "carpal ligament," "transverse carpal," and "proximal phalanx" are frequently encountered and have high TF-IDF scores.

Because TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure used to assess a word's importance in a set of documents, these lemmas stand out. It considers both the frequency of a term occurring within a document (term frequency) and the frequency of a term occurring throughout all documents (inverse document frequency).

In this particular instance:

Term Frequency (TF): Denotes the frequency with which a lemma occurs in a particular field of medicine. Within a specialty, lemmas with high TF scores are those that are frequently encountered.

Inverse Document Frequency (IDF): It measures a lemma's rarity or uniqueness in relation to all medical specialties. Lesmmas with high IDF scores are less prevalent in all specialisations combined.

Lemmas that are both frequently occurring within a particular specialty (high TF) and relatively rare across all specialties (high IDF) are therefore those with high TF-IDF scores. These lemmas are probably very pertinent and typical of that specific field of medicine. Lemmas with high TF-IDF scores within the Orthopaedic specialty—such as "range motion," "carpal ligament," and so on—in the output that has been provided are indicative of their significance and relevance in that field.

```
analysis.data %>% dplyr::select(medical_specialty, transcription) %>% dplyr::filter(stringr::str_detect
```

```
## # A tibble: 1 x 2
##   medical_specialty transcription
##   <chr>             <chr>
## 1 Surgery           preoperative diagnoses hallux rigidus left foot elevated me~
```

```
analysis.data %>%
  dplyr::select(medical_specialty, transcription) %>%
  dplyr::filter(stringr::str_detect(transcription, 'range motion')) %>%
  dplyr::slice(1)
```

9. Extract an example of one of the other "top lemmas" by modifying the above code

```
## # A tibble: 1 x 2
##   medical_specialty transcription
##   <chr>             <chr>
## 1 Surgery           preoperative diagnoses hallux rigidus left foot elevated me~
```

Topic Modelling

First lets calculate a term frequency matrix for each transcription:

```
lemma.counts <- lemmatized.data %>% dplyr::count(note_id, lemma)
total.counts <- lemma.counts %>%
                      dplyr::group_by(note_id) %>%
                      dplyr::summarise(total=sum(n))
```

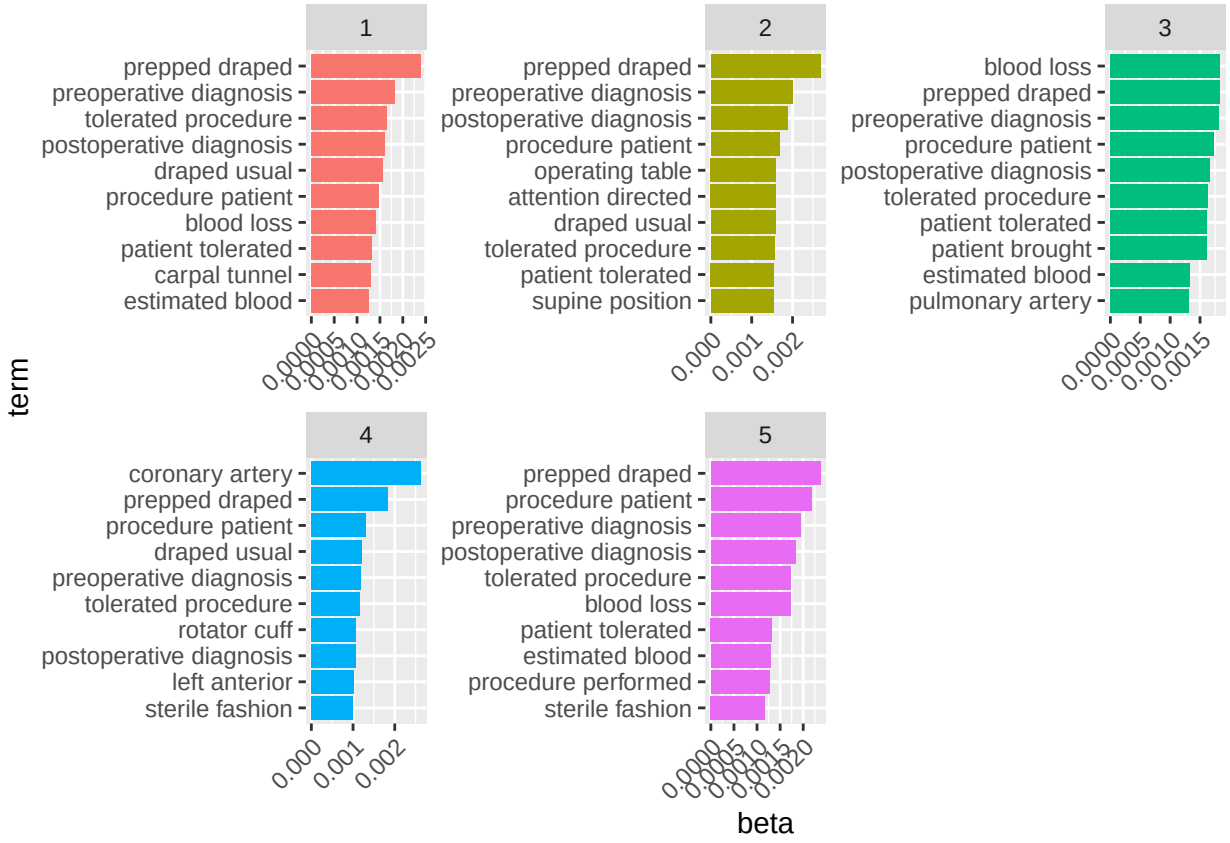
```
all.counts <- dplyr::left_join(lemma.counts, total.counts)
```

```
## Joining with `by = join_by(note_id)`
```

```
emr.dcm <- all.counts %>% tidytext::cast_dtm(note_id, lemma, n)
```

```
emr.lda <- topicmodels::LDA(emr.dcm, k=5, control=list(seed=42))
emr.topics <- tidytext::tidy(emr.lda, matrix='beta')
```

```
top.terms <- emr.topics %>% dplyr::group_by(topic) %>%
  dplyr::slice_max(beta, n=10) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)


top.terms %>%
  dplyr::mutate(term=tidytext::reorder_within(term, beta, topic)) %>%
  ggplot2::ggplot(ggplot2::aes(beta, term, fill=factor(topic))) +
    ggplot2::geom_col(show.legend=FALSE) +
    ggplot2::facet_wrap(~ topic, scales='free')  +
    ggplot2::theme(axis.text.x = element_text(angle = 45,vjust = 1,hjust = 1)) +
    tidytext::scale_y_reordered()
```

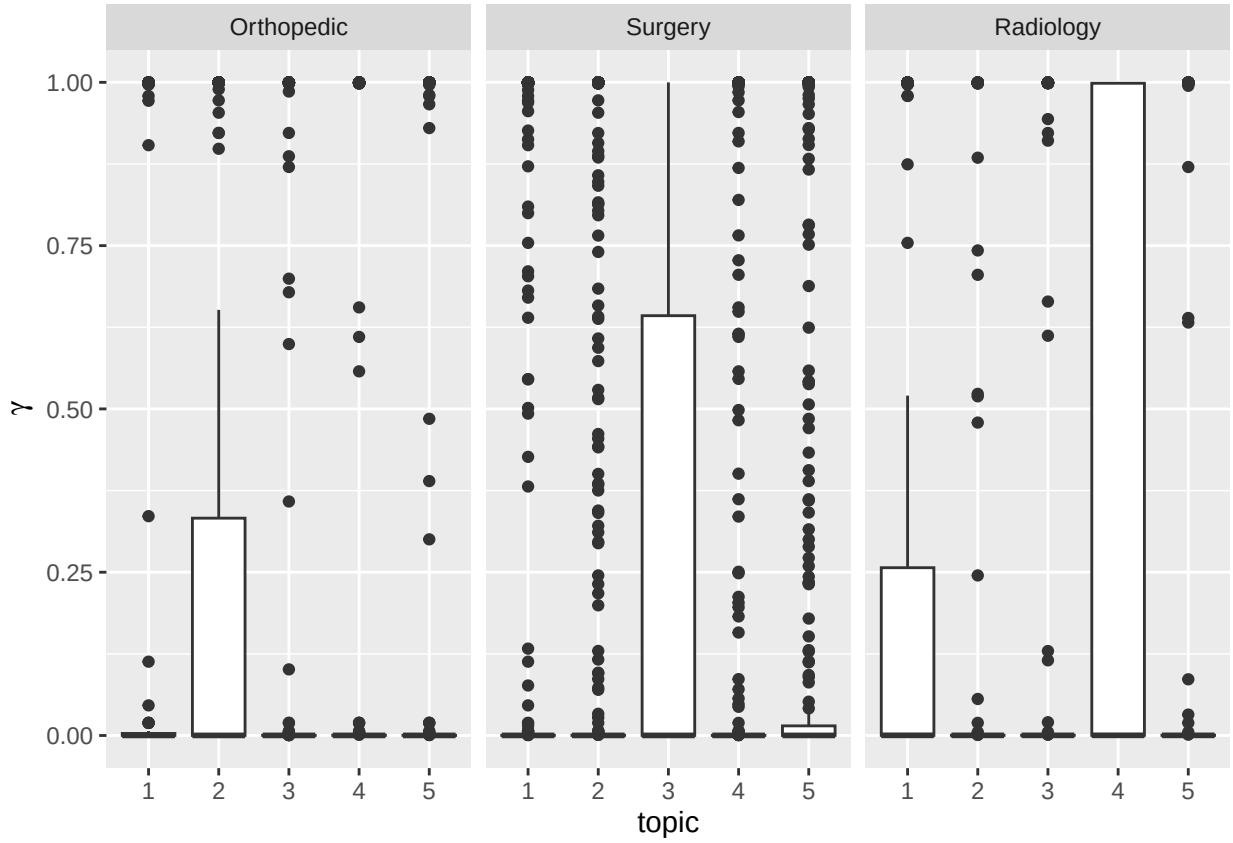



```
specialty_gamma <- tidytext::tidy(emr.lda, matrix='gamma')

# we need to join in the specialty from the note_id
note_id_specialty_mapping <- lemmatized.data %>%
```

```
  dplyr::mutate(document=as.character(note_id)) %>%
  dplyr::select(document, medical_specialty) %>%
  dplyr::distinct()

specialty_gamma <- dplyr::left_join(specialty_gamma, note_id_specialty_mapping)
```

```
## Joining with `by = join_by(document)`
```

```
specialty_gamma %>%
  dplyr::mutate(medical_specialty = reorder(medical_specialty, gamma * topic)) %>%
  ggplot2::ggplot(ggplot2::aes(factor(topic), gamma)) +
  ggplot2::geom_boxplot() +
  ggplot2::facet_wrap(~ medical_specialty) +
  ggplot2::labs(x = "topic", y = expression(gamma))
```




```
# Repeat LDA with 6 topics
emr.lda6 <- topicmodels::LDA(emr.dcm, k=6, control=list(seed=42))
emr.topics6 <- tidytext::tidy(emr.lda6, matrix='beta')

# Top terms per topic (6 topics)
top.terms6 <- emr.topics6 %>%
  dplyr::group_by(topic) %>%
  dplyr::slice_max(beta, n=10) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)
```

```
# Reorder terms within topic
top.terms6 <- top.terms6 %>%
  dplyr::mutate(term=tidytext::reorder_within(term, beta, topic))

# Compare terms between 3-topic and 6-topic models
overlapping_terms <- top.terms %>%
  dplyr::full_join(top.terms6, by="term") %>%
  dplyr::filter(!is.na(beta.x) & !is.na(beta.y))

# View overlapping terms (examine if top terms from 3-topic appear)
overlapping_terms
```

10. Repeat this with a 6 topic LDA, do the top terms from the 3 topic LDA still turn up? How do the specialties get split into sub-topics?

```
## # A tibble: 0 x 5
## # i 5 variables: topic.x <int>, term <chr>, beta.x <dbl>, topic.y <int>,
## #   beta.y <dbl>
```

```
# Visualize topics (6 topics)
top.terms6 %>%
  ggplot2::ggplot(ggplot2::aes(beta, term, fill=factor(topic))) +
    ggplot2::geom_col(show.legend=FALSE) +
    ggplot2::facet_wrap(~ topic, scales='free')  +
    ggplot2::theme(axis.text.x = element_text(angle = 45,vjust = 1,hjust = 1)) +
    tidytext::scale_y_reordered()
```