Practical 2: Electronic Medical Records

Finlay Maguire

2022-06-05

```
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
```

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

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

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

```
library(tidytext)
library(textstem)
```

```
## Loading required package: koRpus.lang.en
```

```
## Loading required package: koRpus
```

```
## Loading required package: sylly
```

```
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
```

```
##
## Attaching package: 'koRpus'
```

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

```
library(clinspacy)
```

```
## 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(topicmodels)
```

This practical is based on exploratory data analysis, named entity recognition, and topic modelling of unstructured medical note free-text data derived from electronic medical records (EMR). Real EMR data is very difficult to access without a specific need/request so this data set is derived from medical transcription data instead. I'll also caveat that the options of natural language processing (NLP) in R are far inferior to those available in Python.

First, install the packages in the setup block (`install.packages(c("readr", "dplyr", "tidyr", "ggplot2", "tidtext", "textstem", "clinspacy", "topicmodels"))`).

```
#also install external dependency: gsl
install.packages(c("topicmodels"))
```

Note: To try and make it clearer which library certain functions are coming from clearer, I'll try to do explicit imports throughout this notebook.

Data Parsing

After that we can grab the dataset directly from the `clinspacy` library.

```
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,~
```

There is no explanation or data dictionary with this dataset, which is a surprisingly common and frustrating turn of events!

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 in this dataset contains.

A: note_id: A unique identifier for the medical note represented by the record entry. description: A brief description of the note's topic. medical_specialty: The department to which the note belongs. sample_name: A title for the note. transcription: The full text of the note. keywords: Search tags for the note.

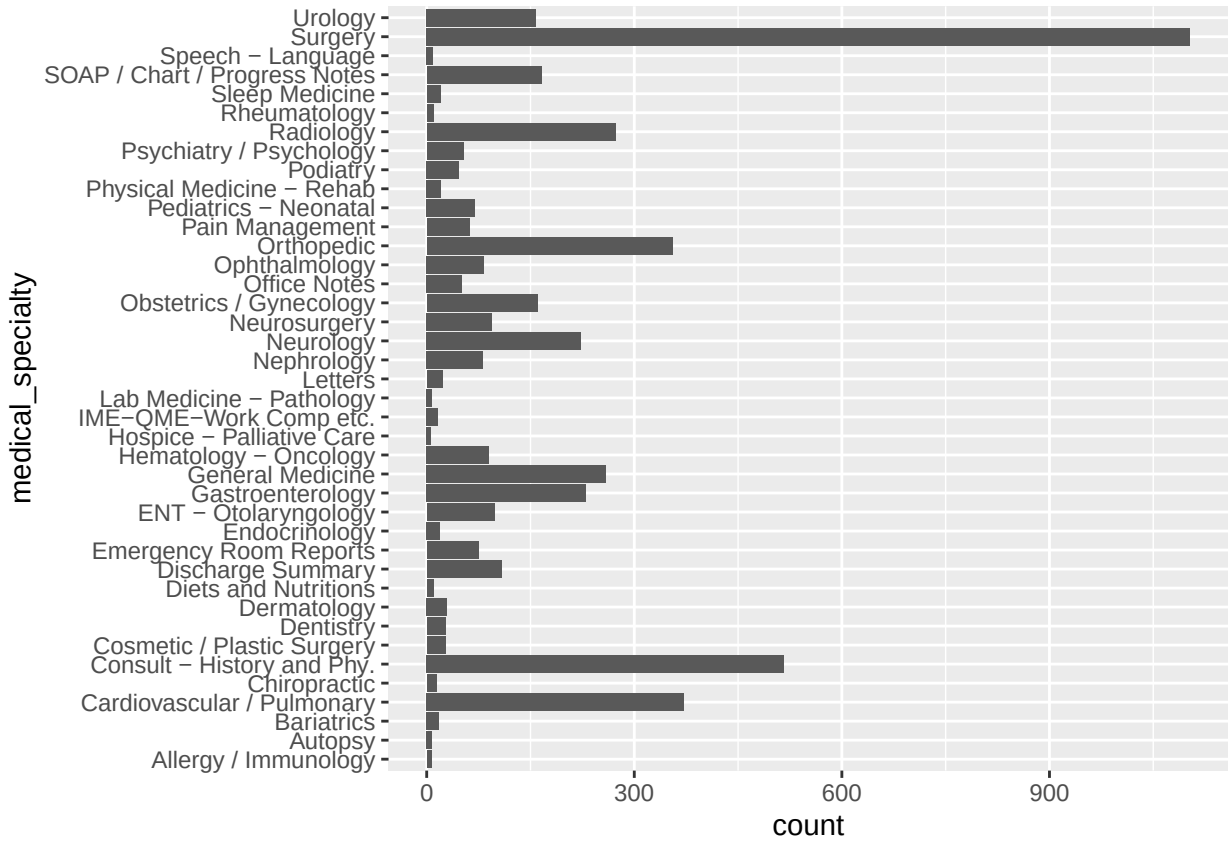
Let's see how many different medical specialties are featured in these notes:

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

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

So, how many transcripts are there from each specialty:

```
ggplot2::ggplot(raw.data, ggplot2::aes(y=medical_specialty)) + ggplot2::geom_bar()
```



Let's make our life easier and filter down to 3 specialties: a diagonstic/lab, a medical, and a surgical specialty

```
analysis.data <- raw.data %>% dplyr::filter(medical_specialty %in% c("Neurology", "Radiology", "Neurosu
```

Text Processing

Let's now apply our standard pre-processing to the transcripts from these specialties.
We are going to use the `tidytext` package to tokenise the transcript free-text.
By default this tokenises to words but other options include characters, n-grams, sentences, lines, paragraphs, or separation around a regular expression.

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

How many unique tokens are there in the transcripts from each specialty:

```
tokenized.data %>% dplyr::group_by(medical_specialty) %>% dplyr::distinct(word) %>% dplyr::summarise(n=
```

```
## # A tibble: 3 x 2
##   medical_specialty     n
##   <chr>             <int>
## 1 Neurology          9015
## 2 Neurosurgery       4263
## 3 Radiology          7313
```

However, there are a lot of extremely common words e.g., "the", "of", "to", and so forth.
These are known as stop words and we can remove them relative easily using a list from `tidytext::stop_words` and `dplyr::anti_join()`

2 How many stop words are there in `tidytext::stop_words`?

A: 534 for Neurology, 410 for Neurosurgery, and 431 for Radiology, totalling 1375 stop words.

```
no.stop.tokenized.data <- tokenized.data %>% dplyr::anti_join(tidytext::stop_words)
```

```
## Joining, by = "word"
```

```
no.stop.tokenized.data %>% dplyr::group_by(medical_specialty) %>% dplyr::distinct(word) %>% dplyr::summa
```

```
## # A tibble: 3 x 2
##   medical_specialty     n
##   <chr>             <int>
## 1 Neurology          8481
## 2 Neurosurgery       3853
## 3 Radiology          6882
```

3 How many unique words are there in each category without stop words and numbers? **A:** There are 8481 unique words for Neurology, 3853 for Neurosurgery, and 6882 for Radiology.

Sometimes we are interested in tokenising/segmenting things other than words like whole sentences or paragraphs.
4 How many unique sentences are there in each category? Hint: use `?tidytext::unnest_tokens` to see the documentation for this function.

```
analysis.data  %>% tidytext::unnest_tokens(sentence,transcription,token='sentences') %>% dplyr::group_by
```

```
## # A tibble: 3 x 2
##   medical_specialty     n
##   <chr>             <int>
## 1 Neurology          6758
## 2 Neurosurgery       2971
## 3 Radiology          4643
```

```
# can also summarise(sentence) to view the sentences in a table alongside medical_speciality
```

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

```
no.stop.tokenized.data %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::count(word, sort = TRUE) %>%
  dplyr::top_n(5)
```

```
## Selecting by n
```

```
## # A tibble: 15 x 3
## # Groups:   medical_specialty [3]
##    medical_specialty word          n
##    <chr>             <chr>     <int>
##  1 Radiology         left        701
##  2 Neurology         left        672
##  3 Neurology         patient     648
##  4 Radiology         normal      644
##  5 Neurology         2           533
##  6 Neurology         normal      485
##  7 Radiology         2           466
##  8 Neurology         history     429
##  9 Radiology         1           409
## 10 Neurosurgery      patient     374
## 11 Radiology         3           328
## 12 Neurosurgery      c5          289
```

```
## 13 Neurosurgery      c6          266
## 14 Neurosurgery      procedure   247
## 15 Neurosurgery      left        222
```

We should lemmatize the tokenized words to prevent over counting of similar words before further analyses. Annoyingly, `tidytext` doesn't have a built-in lemmatizer.

5 Do you think a general purpose lemmatizer will work well for medical data? Why not?

A: I think a medically-focused lemmatizer might perform better for medical data, as there are many medical terms that take on very specific meanings which may be incompatible with more common non-medical usages of the terms. Specifically, there are a wide variety of domain-specific terms which might not be picked up correctly by a general lemmatizer. Additionally, the same problems apply when we consider the matter of typos in the data. If they are to be detected and corrected, a database of medical terms should be used, as many terms and especially their inflected forms will not appear in non-medical dictionaries.

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 <- no.stop.tokenized.data %>% dplyr::mutate(lemma=textstem::lemmatize_words(word))
```

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

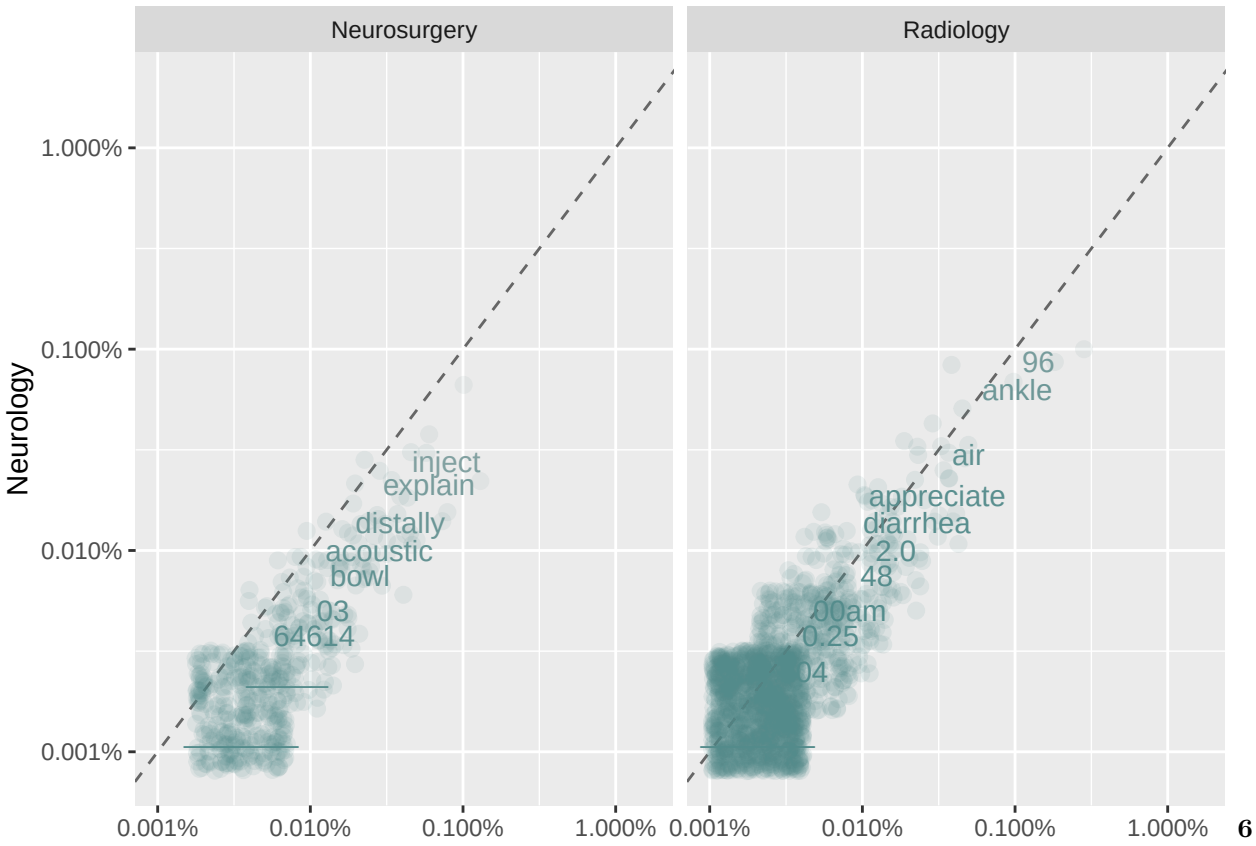
```
lemma.freq <- lemmatized.data %>%
  dplyr::count(medical_specialty, lemma) %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::mutate(proportion = n / sum(n)) %>%
  tidyr::pivot_wider(names_from = medical_specialty, values_from = proportion) %>%
  tidyr::pivot_longer(`Neurosurgery`:`Radiology`,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

And plot the relative proportions

```
ggplot2::ggplot(lemma.freq, ggplot2::aes(x=proportion,
                                         y=`Neurology`,
                                         color=abs(`Neurology` - 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="Neurology", x = NULL)
```

```
## Warning: Removed 26214 rows containing missing values (geom_point).
```

```
## Warning: Removed 26214 rows containing missing values (geom_text).
```



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

A: Few of the lemmas have matching frequencies between Neurosurgery and Neurology, but there are bigger overlaps between Neurology and Radiology. From this, we might predict that Neurosurgery and Radiology would have relatively few term frequencies in common as well, with the biggest overlap remaining for generally clinically relevant terms such as 'patient'. From the contents of the disciplines themselves, I would expect Neurosurgery to be the most unique because of the differing focus in reports that deal with surgeries as opposed to the symptoms and findings of test results which may be common in the other two types of reports.

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

```
lemma.freqSurgery <- 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)
```

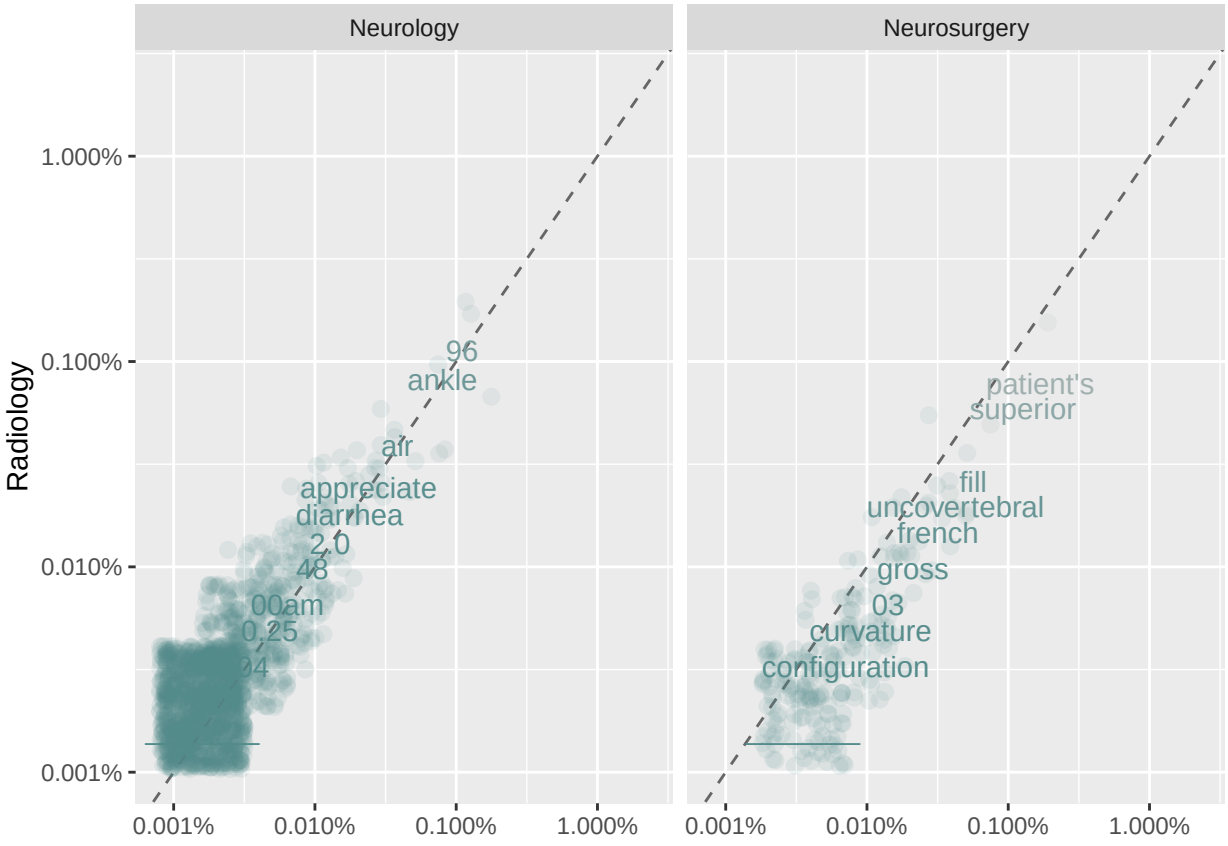
```
lemma.freqSurgery <- lemmatized.data %>%
  dplyr::count(medical_specialty, lemma) %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::mutate(proportion = n / sum(n)) %>%
  tidyr::pivot_wider(names_from = medical_specialty, values_from = proportion) %>%
  tidyr::pivot_longer(`Neurology`:`Neurosurgery`,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

```
ggplot2::ggplot(lemma.freqSurgery, ggplot2::aes(x=proportion,
```

```
                                        y=`Radiology`,
                                        color=abs(`Radiology` - 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="Radiology", x = NULL)
```

```
## Warning: Removed 26486 rows containing missing values (geom_point).
```

```
## Warning: Removed 26486 rows containing missing values (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, by = "medical_specialty"
```

Now we can calculate the term frequency / invariant document frequency (tf-idf):

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

We can then look at the top 10 lemma by tf-idf within each specialty:

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

```
## # A tibble: 31 x 7
## # Groups:   medical_specialty [3]
##    medical_specialty lemma          n total       tf   idf   tf_idf
##    <chr>             <chr>      <int> <int>    <dbl> <dbl>    <dbl>
##  1 Neurology         speech        89 62573 0.00142  0.405 0.000577
##  2 Neurology         93            87 62573 0.00139  0.405 0.000564
##  3 Neurology         impression    86 62573 0.00137  0.405 0.000557
##  4 Neurology         sleep         82 62573 0.00131  0.405 0.000531
##  5 Neurology         b.i.d         30 62573 0.000479 1.10  0.000527
##  6 Neurology         cn            78 62573 0.00125  0.405 0.000505
##  7 Neurology         drug          77 62573 0.00123  0.405 0.000499
##  8 Neurology         hx            70 62573 0.00112  0.405 0.000454
##  9 Neurology         96            69 62573 0.00110  0.405 0.000447
## 10 Neurology         fhx           63 62573 0.00101  0.405 0.000408
## # ... with 21 more rows
```

8 Are there any lemmas that stand out in these lists? Why? **A:** There are a number of lemmas that stand out as being generally nonsensical, such as lemmas '93', and '96', present in both Radiology and Neurology. On the other hand, one lemma that is quite informative is 'speech', within the Neurology notes, where it may act as description of a number of symptoms. Similarly, the 'sleep' lemma in Neurology is informative for the same reasons, and 'impression' within Radiology notes may act as description for radiological findings as well.

We can look at transcriptions using these unusual lemmas to check how they are used with `stringr::str_detect`

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

```
##   medical_specialty
## 1         Radiology
##
## 1 CC:, Episodic monocular blindness, OS.,HX:, This 29 y/o RHF was in her usual healthy state until 2
```

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

```
temp.data <- analysis.data %>% dplyr::select(medical_specialty, transcription) %>% dplyr::filter(stringr
pull(temp.data)
```

```
## [1] "CC: ,BLE weakness and numbness.,HX:, This 59 y/o RHM was seen and released from an ER 1 week pri
```

##A:## What can be seen in this first example of a radiology report containing the substring/lemma "96" is that the context in which 96 occurs is a measurement of heart rate. Looking at other reports, 96 also appears as dates and possibly procedure codes. One circumstance that seems possibly useful judging from the report structure is for seemingly-meaningless data to be picked up from procedure codes, which may be less than ideal given that in this case 96 was not the full token/measurement being reported (instead, 96bpm would have been a more sensible lemma), but is indicative of a type of association that may be useful regardless (i.e., a procedure code itself is a strong indicator and may be detected in the free-text to make decisions about the notes on an individual level).

Topic Modelling

In NLP, we often have collections of documents (in our case EMR transcriptions) that we'd like to divide into groups so that we can understand them separately. Topic modeling is a method for unsupervised classification of such documents, similar to clustering on numeric data.

Latent Dirichlet allocation (LDA) is a particularly popular method for fitting a topic model. It treats each document as a mixture of topics, and each topic as a mixture of words. This allows documents to "overlap" each other in terms of content, rather than being separated into discrete groups, in a way that mirrors typical use of natural language.

- Every document is a mixture of topics. We imagine that each document may contain words from several topics in particular proportions. For example, in a two-topic model we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."
- Every topic is a mixture of words. For example, we could imagine a two-topic model of American news, with one topic for "politics" and one for "entertainment." The most common words in the politics topic might be "President", "Congress", and "government", while the entertainment topic may be made up of words such as "movies", "television", and "actor". Importantly, words can be shared between topics; a word like "budget" might appear in both equally.

LDA is a mathematical method for estimating both of these at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document. There are a number of existing implementations of this algorithm, and we'll explore one of them in depth.

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, by = "note_id"
```

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

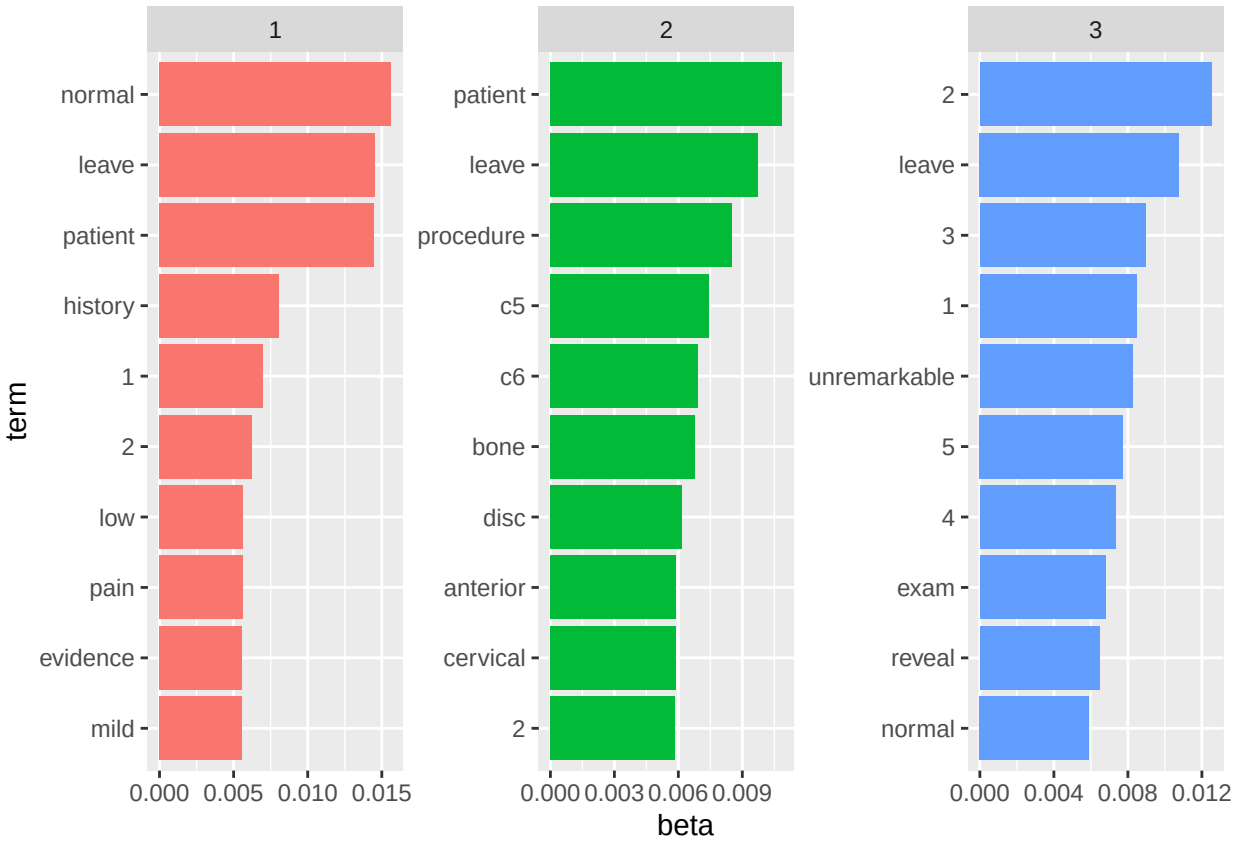
Then we can use LDA function to fit a 3 topic (k=3) LDA-model

```
emr.lda <- topicmodels::LDA(emr.dcm, k=3, control=list(seed=42))
emr.topics <- tidytext::tidy(emr.lda, matrix='beta')
```

Then we can extract the top terms per assigned topic:

```
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')  +
    tidytext::scale_y_reordered()
```



Now we can ask how well do these assigned topics match up to the medical specialties from which each of these transcripts was derived.

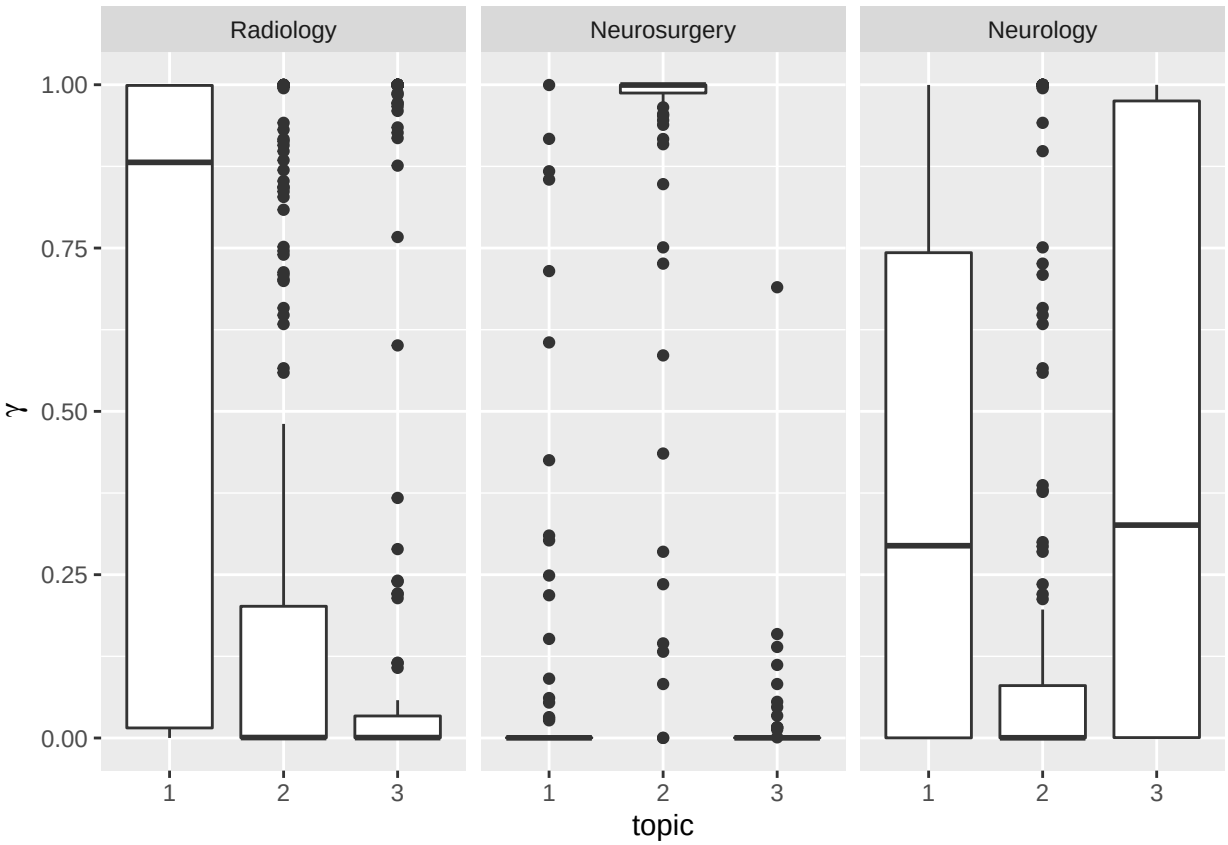
```
specialty_gamma <- tidytext::tidy(emr.lda, matrix='gamma')

# 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, 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))
```



Interestingly, neurosurgery assigns mostly to a single topic but radiology and neurology are both more diverse in transcriptions. We'd possibly expect this from radiology due to referring to imaging for many different diagnoses/reasons. However, this may all just reflect we are using too few topics in our LDA to capture the range of possible assignments.

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?

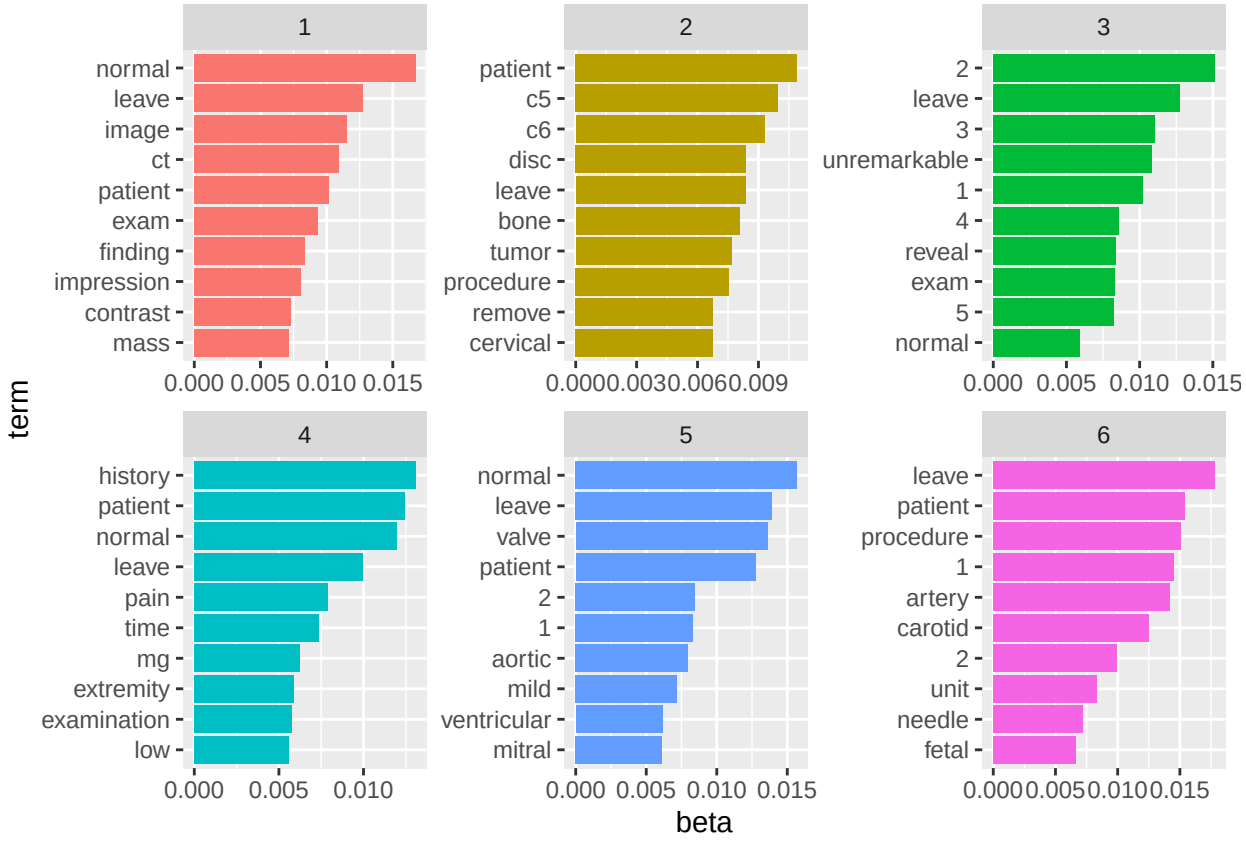
A: The terms of the first topic remain more-or-less intact, and similarly with the second topic although with a notably lower placement for 'procedure'. The third topic also remains with very minor changes. Topic 2 also remains a very good predictor for Neurosurgery, and does a good job of splitting between it and the other two note types. Finally the three new topics perform similarly to topics 1 and 3, helping to provide negative indications against certain disciplines, but not necessarily positively predicting any of them, at least not when considered alone.

```
emr.lda <- topicmodels::LDA(emr.dcm, k=6, 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')  +
```

```
    tidytext::scale_y_reordered()
```



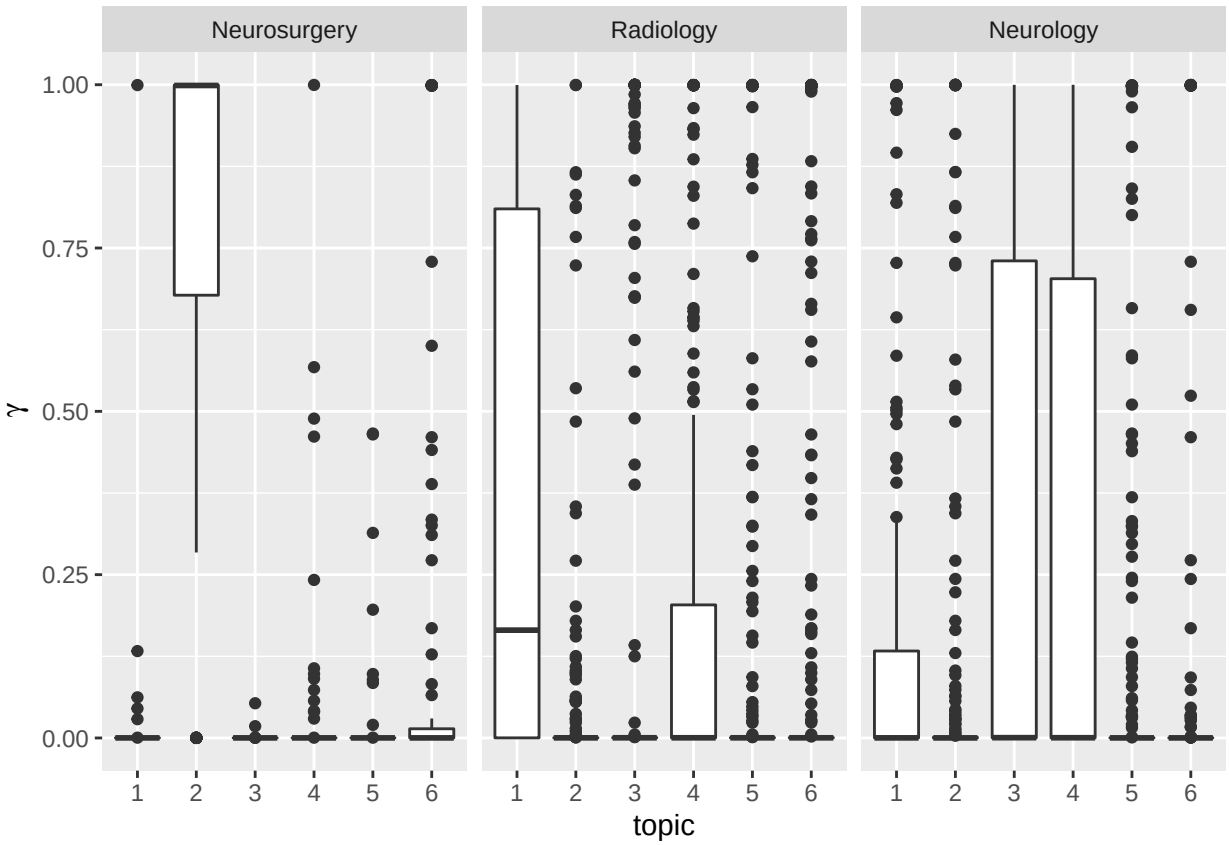

```
specialty_gamma <- tidytext::tidy(emr.lda, matrix='gamma')

# 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, 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))
```



Credits

Examples draw heavily on material (and directly quotes/copies text) from Julia Slige's `tidytext` textbook.