Quantifying Ambiguity

KBC

3/28/2018

## Introduction

Quantitative analyses of aspects of how *clean* the data is, and of how *difficult* it is likely to be from the perspective of understanding its meaning.

### Associated code

The data that is being analyzed here gets generated by:

/Users/transfer/Documents/Scripts-new/ambiguity.measures.pl

\_Actually, I've refactored the whole thing to no longer rely on the Perl script. That makes this a lot easier to keep track of, but it means that the whole thing will have to be run locally.

## Read in data

*Input:* String containing the path to a corpus

*Output:* A tibble with one document per row.

# Note that this function reads in an entire corpus, which means that the resulting data can only be used to analyze the entire corpus. That's good for a lot of things, but for some other things, we will need to process one file at a time.  
read.in.corpus <- function(my.path) {  
my.data <- list.files(path = my.path, pattern = "txt$") %>%   
 map\_chr(~ read\_file(paste(my.path, ., sep = ""))) %>%   
 data\_frame(line = 1, text = .)  
return(my.data)  
}  
data.raw <- read.in.corpus("/Users/transfer/Dropbox/a-m/Corpora/craft-2.0/articles/txt/")  
#typeof(data.raw)  
head(data.raw)

## # A tibble: 6 x 2  
## line text   
## <dbl> <chr>   
## 1 1. "Intraocular pressure in genetically distinct mice: an update and…  
## 2 1. "BRCA2 and homologous recombination\n\nAbstract\n\nTwo recent pap…  
## 3 1. "Cloning and characterization of the mouse Mcoln1 gene reveals an…  
## 4 1. "Embryonic stem cells and mice expressing different GFP variants …  
## 5 1. "Morphological characterization of the AlphaA- and AlphaB-crystal…  
## 6 1. "Brn3c null mutant mice show long-term, incomplete retention of s…

## Produce counts

# I \*think\* the expected input is a data frame  
get.counts <- function(data.raw) {  
counts <- data.raw %>%  
 unnest\_tokens(word, text) %>%  
 count(word, sort = TRUE)  
  
return(counts)  
}  
counts <- get.counts(data.raw)  
head(counts)

## # A tibble: 6 x 2  
## word n  
## <chr> <int>  
## 1 the 23093  
## 2 of 16782  
## 3 and 14231  
## 4 in 13943  
## 5 to 7511  
## 6 a 6916

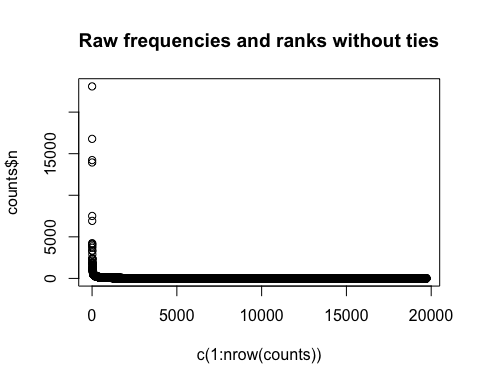
One useful thing to do with the raw frequencies (counts) is to check for words in your document templates (such as fields for the patient name or for the History and Physical) that you might want me to remove from the analysis.

## Basic distributional characteristic: frequency/rank relationship

For any sufficiently large sample of language, relationship between frequency and frequency-ordered rank should be Zipfian.

The reverse does not *necessarily* hold--seeing a Zipfian relationship does *not* necessarily imply that the sample size is "sufficiently large."

plot.frequency.rank.relationship <- function(counts) {  
 plot(c(1:nrow(counts)), counts$n,  
 main = "Raw frequencies and ranks without ties")  
}  
plot.frequency.rank.relationship(counts)



## This is where you look for the rate of spelling errors...

If "data quality" means something like "how much garbage is there in my data?", then one estimate of that would be the incidence of spelling errors. Here we estimate that incidence by:

1. Finding the words that only occur once
2. Removing any words that contain numbers (often lab values and the like) or punctuation (similar)
3. Run the remaining words through a spell checker

#### Why only look at words that only occur once?

By looking at words that only occur once, we make sure that local abbreviations, slang, technical terms, etc. do not get mistakenly flagged by the spell checker. We then eliminate things that look like lab values, physiological measurements, etc. because those are not spelling errors, as such (although they do raise separate issues).

*John, I can make this print out for you without the code for the purpose of putting a presentation together, or I can make it print out the figures into individual files to plug into a PowerPoint presentation, or you can edit the code out manually in Word--whatever your prefer.*

# input: a tibble with two columns: word and n, where n is the count for that word  
estimate.rate.of.spelling.errors <- function(counts) {  
counts  
   
hapax <- counts %>% filter(n == 1)  
head(hapax)  
print(paste("Words in corpus that only occur once:", nrow(hapax)))  
hapax.filtered <- hapax %>% filter(str\_detect(word, "^[\\d\\.\\-\\,]+$") == FALSE)  
print(paste("...after filtering out many numbers:", nrow(hapax.filtered)))  
hapax.filtered  
  
hapax.nonumbers <- hapax %>% filter(str\_detect(word, "\\d") == FALSE)  
print(paste("...after filtering out all words containing numbers:", nrow(hapax.nonumbers)))  
hapax.nonumbers  
  
hapax.with.punctuation <- hapax.nonumbers %>% filter(str\_detect(word, "[\\.\\,\\:\\-\\;]"))  
print(paste("How many of those contain punctuation?", nrow(hapax.with.punctuation)))  
hapax.with.punctuation  
  
hapax.without.punctuation <- hapax.nonumbers %>% filter(str\_detect(word, "[\\.\\,\\:\\-\\;]") == FALSE)  
print(paste("After removing punctuation, too:", nrow(hapax.without.punctuation)))  
  
#hunspell\_check(hapax.without.punctuation$word, dict = dictionary("en\_US"))  
hapax.without.punctuation.spellchecks <- hunspell\_check(hapax.without.punctuation$word, dict = dictionary("en\_US"))  
  
#hapax.without.punctuation.spellchecks <- as\_tibble(hapax.without.punctuation.spellchecks)  
  
# TODO do this again, but with mutate()  
hapax.without.punctuation <- as\_tibble(hapax.without.punctuation)  
hapax.without.punctuation <- mutate(hapax.without.punctuation, spellcheck = hapax.without.punctuation.spellchecks)  
  
hapax.without.punctuation  
  
hapax.spelling.ok <- hapax.without.punctuation %>% filter(spellcheck == TRUE)  
hapax.spelling.ok  
hapax.spelling.bad <- hapax.without.punctuation %>% filter(spellcheck == FALSE)  
hapax.spelling.bad  
  
# the estimated rate of spelling errors is the number of words that only occur once AND don't contain numbers or punctuation AND fail a spelling check, divided by the total count of word tokens, i.e. words in the document collection. So:  
  
total.tokens <- sum(counts$n)  
estimated.frequency.of.errors <- (nrow(hapax.spelling.bad) / total.tokens) \* 100  
print(paste("Estimated percentage of spelling errors in this data:", estimated.frequency.of.errors))  
  
return(estimated.frequency.of.errors)  
  
#hapax.without.punctuation %>% mutate(spellcheck.result, hunspell\_check(hapax.without.punctuation$word, dict = dictionary("en\_US")))  
  
#hapax.without.punctuation.spellchecks  
#hapax.spelling.errors <- hapax.without.punctuation.spellchecks %>% filter(value == FALSE)  
#print(paste("Number of errors after removing all words containing numbers or punctuation:"), length(hapax.spelling.errors)))  
  
} # close function definition estimate.rate.of.spelling.errors()  
#estimate.rate.of.spelling.errors(counts.craft)  
#estimate.rate.of.spelling.errors(counts.mimic)  
#estimate.rate.of.spelling.errors(counts.mimic.radiology)  
#estimate.rate.of.spelling.errors(counts.mimic.nursing)  
estimate.rate.of.spelling.errors(counts)

## [1] "Words in corpus that only occur once: 7713"  
## [1] "...after filtering out many numbers: 6843"  
## [1] "...after filtering out all words containing numbers: 5634"  
## [1] "How many of those contain punctuation? 107"  
## [1] "After removing punctuation, too: 5527"  
## [1] "Estimated percentage of spelling errors in this data: 0.712857058358326"

## [1] 0.7128571

There are other things that we can do to estimate the quality of this data, as well. For example, we've been looking at spelling errors so far, but the incidence of words that only occur once and are spelled *correctly* is also an index of how difficult this data will be to work with. You could think of that as the likelihood of having things in your real data that were not in the training data. This is an important variable in understanding how well machine learning is likely to work for your data: the algorithm can only build a good model for things for which it has enough instances in the training data. Another thing that we could take into account is how many of those things looked like lab values, physiological measurements, etc.--things like that can be normalized into usable features, which can *increase* performance. But, you get the flavor of how this part of the analysis works.

## How hard will it be to extract information from this?

Now that we have an estimate of how much garbage there is in the data, we would like to know how difficult it is likely to be to work with this data. We will estimate *that* by seeing how common some phenomena that we know cause problems in any language processing task are in our data.

quantify.negation <- function(data.tibble) {  
negatives <- c("no", "not", "none", "denies", "nothing") # anything? any?  
  
counts.negatives <- data.tibble %>% filter(word %in% negatives) %>%  
 count(word, sort = TRUE)   
counts.negatives   
}

quantify.prepositional.phrases <- function(data.tibble) {  
prepositions <- c("of", "to", "by", "with", "from", "for")  
counts.prepositions <- data.tibble %>% filter(word %in% prepositions) %>% count(word, sort = TRUE)  
counts.prepositions  
}

quantify.anaphoric.reference <- function(data.tibble) {  
anaphora <- c("i", "me", "my", "we", "us", "our", "you", "your",   
 "he", "him", "his", "she", "her", "they", "them", "their")  
counts.anaphora <- data.tibble %>% filter(word %in% anaphora) %>% count(word, sort = TRUE)  
counts.anaphora  
}

quantify.conjunctions <- function(data.tibble) {  
conjunctions <- c("and", "or", "but")  
counts.conjunctions <- data.tibble %>% filter(word %in% conjunctions) %>% count(word, sort = TRUE)  
counts.conjunctions  
# TODO: turn these counts into frequencies  
# TODO: need to return something. Maybe break this up into separate functions??  
}

# TODO: I could save some memory by doing this earlier, and then using the resulting object to get the counts, instead of running through unnest\_tokens() twice.   
# Oh, that might not be true--it depends on whether or not we need individual files, e.g. per patient. So far, we haven't. But, for things like calculating type/token ratio, we will...  
  
quantify.ambiguity <- function(data.raw) {  
#Note that tibble is a kind of R data structure that is nice for doing quanitative analyses of textual data.  
data.tibble <- data.raw %>%  
 unnest\_tokens(word, text)   
  
data.tibble  
return(data.tibble)  
} # close function definition quantify.ambiguity()  
data.tibble <- quantify.ambiguity(data.raw)  
quantify.negation(data.tibble)

## # A tibble: 3 x 2  
## word n  
## <chr> <int>  
## 1 not 1610  
## 2 no 617  
## 3 none 31

quantify.anaphoric.reference(data.tibble)

## # A tibble: 13 x 2  
## word n  
## <chr> <int>  
## 1 we 1758  
## 2 our 419  
## 3 their 408  
## 4 i 348  
## 5 they 243  
## 6 us 64  
## 7 them 58  
## 8 his 13  
## 9 me 8  
## 10 he 5  
## 11 her 4  
## 12 him 1  
## 13 she 1

quantify.conjunctions(data.tibble)

## # A tibble: 3 x 2  
## word n  
## <chr> <int>  
## 1 and 14231  
## 2 or 1829  
## 3 but 782

quantify.prepositional.phrases(data.tibble)

## # A tibble: 6 x 2  
## word n  
## <chr> <int>  
## 1 of 16782  
## 2 to 7511  
## 3 for 4239  
## 4 with 4144  
## 5 by 2897  
## 6 from 2341

## Quantifying speculation

One indicator of how much you will be able to weight information mined from your text is how the writers express their confidence in their statements. Quantifying confidence is complicated, but there is a lot known about how to do it. For now, we will just estimate it by calculating the frequencies of some lexical cues to speculation, such as *rule out, possible,* and the like.

data.tibble <- data.raw %>%  
 unnest\_tokens(word, text)   
  
data.tibble

## # A tibble: 456,473 x 2  
## line word   
## <dbl> <chr>   
## 1 1. intraocular  
## 2 1. pressure   
## 3 1. in   
## 4 1. genetically  
## 5 1. distinct   
## 6 1. mice   
## 7 1. an   
## 8 1. update   
## 9 1. and   
## 10 1. strain   
## # ... with 456,463 more rows

speculation.cues <- c("rule", "R/O", "possible", "eventual") # note that R/O may get broken up in earlier processing, "rule out" won't actually be found if you're just looking at single words, "could be" is a clear speculation cue, but requires that you look at two-word sequences--these issues can be coped with. For the moment, we'll just look at the general concept.  
data.tibble %>% filter(word %in% speculation.cues) %>% count(word, sort = TRUE)

## # A tibble: 3 x 2  
## word n  
## <chr> <int>  
## 1 possible 153  
## 2 rule 19  
## 3 eventual 4

### Now let's look at what kinds of functions some of these things play. Prepositions can have many functions *and* many meanings, so let's look at some of those. Note that this will have to be a manual analysis.

get.trigrams <- function(data.raw) {  
 my.trigrams <- data.raw %>%   
 unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%  
 separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%  
 #filter(word2 == "of") %>%  
 count(word1, word2, word3, sort = TRUE)  
 my.trigrams  
 return(my.trigrams)  
}  
trigrams.all <- get.trigrams(data.raw)  
trigrams.all

## # A tibble: 338,087 x 4  
## word1 word2 word3 n  
## <chr> <chr> <chr> <int>  
## 1 the absence of 189  
## 2 the presence of 187  
## 3 pgc 1α mice 186  
## 4 wild type and 179  
## 5 in situ hybridization 167  
## 6 as well as 155  
## 7 data not shown 152  
## 8 in the absence 132  
## 9 mcm4 6 7 131  
## 10 the expression of 125  
## # ... with 338,077 more rows

filter.trigrams <- function(trigrams, string.to.match) {  
 filtered.trigrams <- trigrams %>%  
 filter(word2 == string.to.match) %>%  
 count(word1, word2, word3, sort = TRUE)  
  
 return(filtered.trigrams)  
}  
filtered.trigrams <- filter.trigrams(trigrams.all, "of")  
filtered.trigrams

## # A tibble: 10,797 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 0.005 of the 1  
## 2 0.01 of ark 1  
## 3 0.01 of the 1  
## 4 0.1 of total 1  
## 5 0.2μm of each 1  
## 6 1 of itself 1  
## 7 1 of myosin 1  
## 8 1 of p53 1  
## 9 1 of the 1  
## 10 1 of trip13gt 1  
## # ... with 10,787 more rows

## Going through a corpus of clinical data

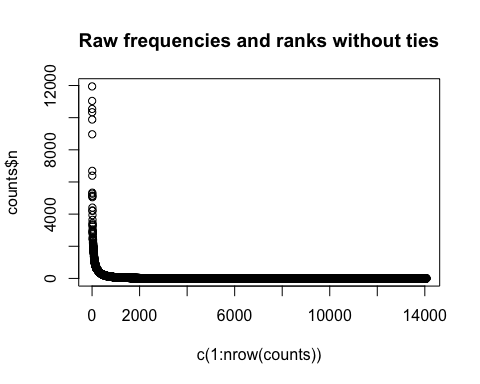
With that background, let's see how we can go through a single corpus of clinical data.

data.raw <- read.in.corpus("/Users/transfer/Dropbox/a-m/Corpora/MIMIC2/md/")

counts <- get.counts(data.raw)  
head(counts)

## # A tibble: 6 x 2  
## word n  
## <chr> <int>  
## 1 with 11945  
## 2 to 11040  
## 3 in 10542  
## 4 and 10339  
## 5 on 9884  
## 6 no 8966

plot.frequency.rank.relationship(counts)



estimate.rate.of.spelling.errors(counts)

## [1] "Words in corpus that only occur once: 6153"  
## [1] "...after filtering out many numbers: 5154"  
## [1] "...after filtering out all words containing numbers: 4282"  
## [1] "How many of those contain punctuation? 168"  
## [1] "After removing punctuation, too: 4114"  
## [1] "Estimated percentage of spelling errors in this data: 0.436816644803723"

## [1] 0.4368166

quantify.anaphoric.reference(data.tibble)

## # A tibble: 13 x 2  
## word n  
## <chr> <int>  
## 1 we 1758  
## 2 our 419  
## 3 their 408  
## 4 i 348  
## 5 they 243  
## 6 us 64  
## 7 them 58  
## 8 his 13  
## 9 me 8  
## 10 he 5  
## 11 her 4  
## 12 him 1  
## 13 she 1

quantify.conjunctions(data.tibble)

## # A tibble: 3 x 2  
## word n  
## <chr> <int>  
## 1 and 14231  
## 2 or 1829  
## 3 but 782

quantify.prepositional.phrases(data.tibble)

## # A tibble: 6 x 2  
## word n  
## <chr> <int>  
## 1 of 16782  
## 2 to 7511  
## 3 for 4239  
## 4 with 4144  
## 5 by 2897  
## 6 from 2341

trigrams.all <- get.trigrams(data.raw)

filtered.trigrams <- filter.trigrams(trigrams.all, "of")  
filtered.trigrams

## # A tibble: 2,257 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 1 of 5 1  
## 2 1 of countdown 1  
## 3 10 of amp 1  
## 4 10 of apnea 1  
## 5 10 of countdown 1  
## 6 10 of erythro 1  
## 7 10 of gent 1  
## 8 100.4 of note 1  
## 9 1012 of hospital1 1  
## 10 11 of a 1  
## # ... with 2,247 more rows

filtered.trigrams <- filter.trigrams(trigrams.all, "to")  
filtered.trigrams

## # A tibble: 3,846 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 0.2 to 2.0 1  
## 2 0.6 to go 1  
## 3 1 to 50 1  
## 4 1 to 80s 1  
## 5 1 to aid 1  
## 6 10 to 20 1  
## 7 10 to 2400 1  
## 8 10 to be 1  
## 9 10 to receive 1  
## 10 100 to maintain 1  
## # ... with 3,836 more rows

filtered.trigrams <- filter.trigrams(trigrams.all, "by")  
filtered.trigrams

## # A tibble: 491 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 0.5cm by md 1  
## 2 106 by dr 1  
## 3 17 by rrt 1  
## 4 19 by ambulance 1  
## 5 191 by decreasing 1  
## 6 192 by first 1  
## 7 199 by telephone 1  
## 8 20 by gavage 1  
## 9 3 by l 1  
## 10 32 by ua 1  
## # ... with 481 more rows

filtered.trigrams <- filter.trigrams(trigrams.all, "for")  
filtered.trigrams

## # A tibble: 2,451 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 0.3 for which 1  
## 2 1 for echo 1  
## 3 1 for this 1  
## 4 10 for brady 1  
## 5 10 for this 1  
## 6 100 for prolonged 1  
## 7 1018 for the 1  
## 8 1099 for sepsis 1  
## 9 11 for this 1  
## 10 115 for 24 1  
## # ... with 2,441 more rows

filtered.trigrams <- filter.trigrams(trigrams.all, "with")  
filtered.trigrams

## # A tibble: 4,756 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 00 with interpretter 1  
## 2 0430 with results 1  
## 3 1 with 2 1  
## 4 1 with effect 1  
## 5 1 with feed 1  
## 6 1 with feeding 1  
## 7 1 with good 1  
## 8 1 with hx 1  
## 9 1 with improvement 1  
## 10 1 with last 1  
## # ... with 4,746 more rows

filtered.trigrams <- filter.trigrams(trigrams.all, "from")  
filtered.trigrams

## # A tibble: 452 x 4  
## word1 word2 word3 nn  
## <chr> <chr> <chr> <int>  
## 1 0.1 from 1.9 1  
## 2 0.2 from 3.8 1  
## 3 0.2 from 4.9 1  
## 4 0.2 from 5.1 1  
## 5 0.25 from day 1  
## 6 0.3 from 3.8 1  
## 7 0.3 from 4.2 1  
## 8 0.3 from 4.9 1  
## 9 0.3 from 5.9 1  
## 10 0.3 from 6.8 1  
## # ... with 442 more rows

*Need to pull out speculation into a function.*

# JOHN, YOU CAN REMOVE ANYTHING AFTER THIS HEADING BEFORE SHARING

#data.mimic <- "My dog is a very bad dog. I love her terribly. She loves me, too. She is very jealous of me. She likes to go for walks with me."  
  
# PHYSICIAN NOTES  
data.mimic <- read\_file("/Users/transfer/Dropbox/a-m/Corpora/MIMIC2/mimic2\_500K.txt")  
data.mimic <- data\_frame(line = 1, text = data.mimic)  
counts.mimic <- data.mimic %>%  
 unnest\_tokens(word, text) %>%  
 count(word, sort = TRUE)  
  
plot(c(1:nrow(counts.mimic)), counts.mimic$n)  
counts.mimic  
  
# NURSING NOTES  
path.mimic.nursing <- "/Users/transfer/Dropbox/a-m/Corpora/MIMIC2/nursing/"  
data.mimic.nursing <- list.files(path = path.mimic.nursing, pattern = "") %>%   
 map\_chr(~ read\_file(paste(path.mimic.nursing, ., sep = ""))) %>%   
 data\_frame(line = 1, text = .)  
counts.mimic.nursing <- data.mimic.nursing %>%  
 unnest\_tokens(word, text) %>%  
 count(word, sort = TRUE)  
  
plot(c(1:nrow(counts.mimic.nursing)), counts.mimic.nursing$n)  
counts.mimic.nursing  
  
# RADIOLOGY REPORTS  
# NURSING NOTES  
path.mimic.radiology <- "/Users/transfer/Dropbox/a-m/Corpora/MIMIC2/radiology/kev\_clinical\_radiology.txt"  
#data.mimic.radiology <- list.files(path = path.mimic.radiology, pattern = "") %>%   
 #map\_chr(~ read\_file(paste(path.mimic.radiology, ., sep = ""))) %>%   
# data\_frame(line = 1, text = .)  
  
data.mimic.radiology <- read\_file(path.mimic.radiology)  
data.mimic.radiology <- data\_frame(line = 1, text = data.mimic.radiology)  
counts.mimic.radiology <- data.mimic.radiology %>%  
 unnest\_tokens(word, text) %>%  
 count(word, sort = TRUE)  
  
plot(c(1:nrow(counts.mimic.radiology)), counts.mimic.radiology$n)  
counts.mimic.radiology  
  
  
# CRAFT CORPUS  
path.craft <- "/Users/transfer/Dropbox/a-m/Corpora/craft-2.0/articles/txt/"  
data.craft <- list.files(path = path.craft, pattern = "") %>%   
 map\_chr(~ read\_file(paste(path.craft, ., sep = ""))) %>%   
 data\_frame(line = 1, text = .)  
counts.craft <- data.craft %>%  
 unnest\_tokens(word, text) %>%  
 count(word, sort = TRUE)  
  
plot(c(1:nrow(counts.craft)), counts.craft$n)  
counts.craft

## What are the most common prepositions?

counts.mimic  
counts.mimic.nursing  
counts.mimic.radiology  
counts.craft

...so, it looks like the most common preposition is *of*.

## Looking at *of* in the various corpora

We want to know what kinds of things can take *of* as a prepositional phrase modifier, and what kinds of things can be the nouns in those prepositional phrases. We could do that with Sketch Engine, but let's try just looking at c ollocations and adjacent words and see how far that gets us. If nothing else, it might give us some hints about what kinds of things to focus on in Sketch Engine. In the best-case scenario, it will give us enough things to make clear what kinds of relationships we should be looking at in the qualitative analysis.

So: let's start by getting the common trigrams. That gets us the thing being modified (modulo a lot of noise from cases where it's actually attached to a verb that's too far in front of it for us to see it) and the thing in the prepositional phrase.

data.mimic %>%   
 unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%  
 separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%  
 filter(word2 == "of") %>%  
 count(word1, word2, word3, sort = TRUE)

In the discharge summaries, the most frequent use of *of* is in temporal expressions: *date of birth*, *day of life* (the patients are pediatric), *time of discharge*, *day of admission*, *months of age*, and *day of discharge*.

data.mimic.nursing %>%   
 unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%  
 separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%  
 filter(word2 == "of") %>%  
 count(word1, word2, word3, sort = TRUE)

In contrast, *of* has much more diverse functions in the nursing notes---no single function dominates, as temporal expressions do in the physicians' discharge notes. Document sections (function = structuring the discourse): *plan of care*, *review of systems*, a temporal expression *day of life*, modifying a transparent noun (not listed in Merriam-Webster!) *amounts of thick*, *amts of thick*...

data.mimic.radiology %>%   
 unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%  
 separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%  
 filter(word2 == "of") %>%  
 count(word1, word2, word3, sort = TRUE)

Need to figure out the radiology ones...

In CRAFT, on the other hand, the predominant function is that of indicating an argument of a preceding nominalization (Merriam-Webster's 9(b)): *expression of the*, *analysis of the*, *disruption of the*.

data.craft %>%   
 unnest\_tokens(trigram, text, token = "ngrams", n = 3) %>%  
 separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%  
 filter(word2 == "of") %>%  
 count(word1, word2, word3, sort = TRUE)