Analysis of Corpus for Data Science Capstone

Chuck Stietzel

1/10/2017

Synopsis

This is the milestone report for the Data Science Specialization Capstone. The project culminates in the delivery of a Shiny Application that predicts the next word in a phrase that the user enters. A language model will be developed using 3 sources of data, Twitter Tweets, Blog entries, and News items.

This report summarizes the loading and processing of the data and provides an exploratory analysis of the corpus including the frequency of words and n-grams.

Data Loading and Preprocessing

The data provided is in multiple languages, but this study will only consider the English version of the test data. Foreign language and profanity have been ignored at this stage due to low frequency of observation relative to the size of the corpus. Profanity will be filtered out of the model at prediction time.

Many steps were undertaken to clean and normalize the data. This cleansing took the form of substitutions using regular expressions. The preprocessing steps are very computationally intensive, so the code was parallelized to take maximum advantage of the processors available.

First, a conversion of special characters from UTF-8 format into all lowercase ASCII was performed. This allowed for the application of simplified regular expressions. URLs, Hashtags and Re-tweets were deleted from all text and hyphens and underscored were converted to spaces.

Numerical values have been removed. Initially numerical values were replaced with a special string "**num**" so they could be used in the predictions. However, this had the effect of having may n-grams with numbers appear with the highest frequencies.

Many contractions were observed, so they were expanded into their fully worded counterparts. Wherever possible contractions written without apostrophes were identified and replaced as this is often the case in Tweets. Contractions that spelled English words when written without the apostrophes were ignored.

Many abbreviations were converted into their fully worded counterparts as well. These are often used in Tweets and Blogs. (E.g. "n" or "&" -> "and", "b/c" -> "because", or "@" -> "at")

Finally, any letters that are repeated three or more times were replaced by only one instance of the character. So "yeeeeesssssss" becomes "yes". There are times when this would not be desirable but we will (e.g. "coooool"), however, this was ignored.

Tokenization and Corpus Anaylysus

The analysis makes heavy use of the *tidytext* package for tokenization. The procedures run fast and therefore an analysis was run on the entire corpus.

Table 1 provides a summary of the different sources. While there are over 2 million Tweets the number of raw tokens obtained is less than that obtained from News and Blog source due to the 140 character limit on a Tweet. The Blog source has the fewest excerpts to process, but its excerpts fare are the longest on average.

Looking at the number of unique unigrams produced, it is interesting to note that the number of unique "words" in the raw data is much higher in the Twitter source while the News source has the least. This is

probably due to all of the abbreviations and slang in the Twitter data. Blogs being less formal than News sources, also contained a higher number of unique unigrams.

After cleaning and normalizing the data the number of unique unigrams is reduced in all sources while the number of tokens was relatively constant. The unigrams in the Twitter source was reduced by 25%. The compression is probably due to the contraction and abbreviation processing. News unigram counts were reduced by 15%, and Blogs were least affected with a 7% reduction. The News source was affected most by the removal of numbers where dates and counts are often included in the excerpts.

Table 1: Overview of Corpus

		Raw		Cleaned	
Source	Excerpts	Tokens	Unigrams	Tokens	Unigrams
News	1,010,242	34,762,395	284,533	34,214,978	244,060
Blogs	899,288	37,546,246	320,005	37,598,933	300,617
Twitter	2,360,148	30,093,369	370,386	30,180,420	279,886

Sampling and N-Gram Analysis

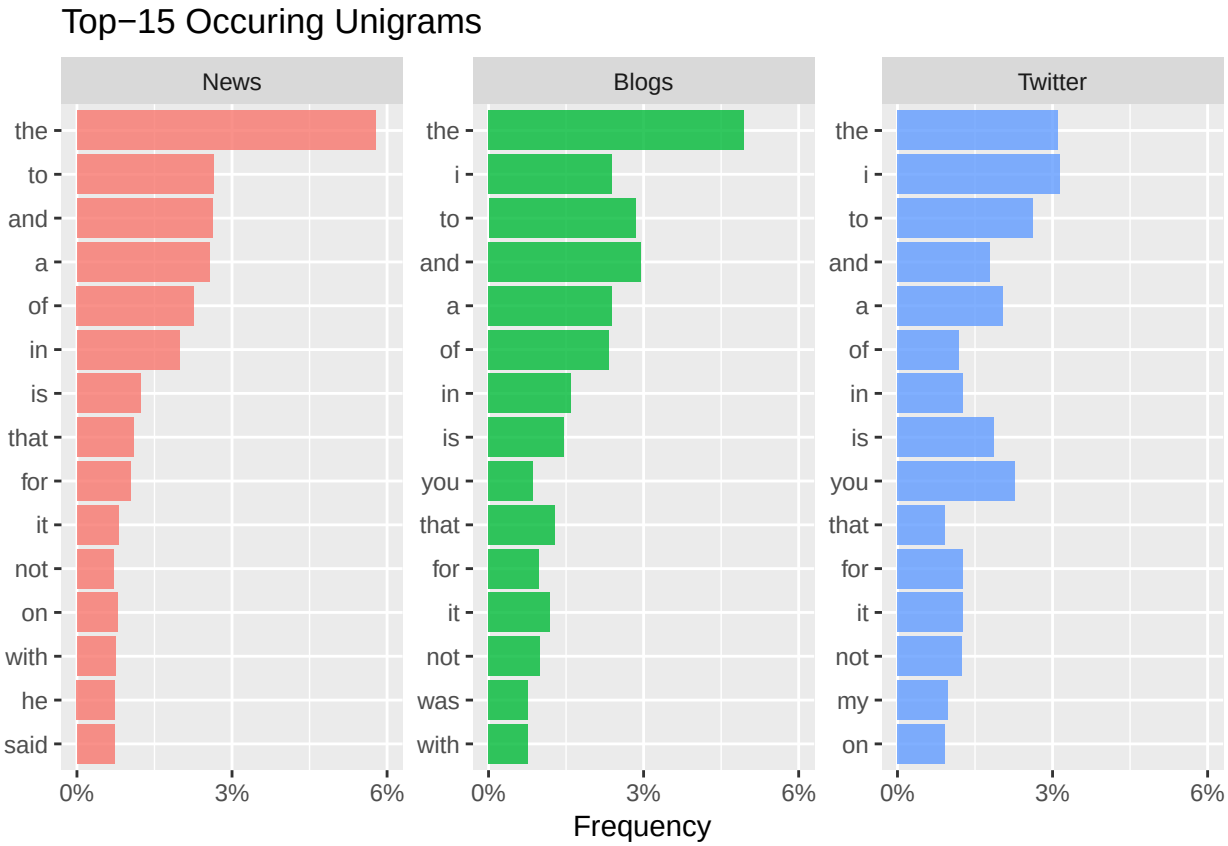
The cleaned corpus was sampled to create partitions for training and testing. 60% of the excerpts from each source will be used to train the prediction models. The remaining 40% will be used to test the models. This set may be further sampled to provide a set for model selection and a set for final out-of-sample accuracy estimation.

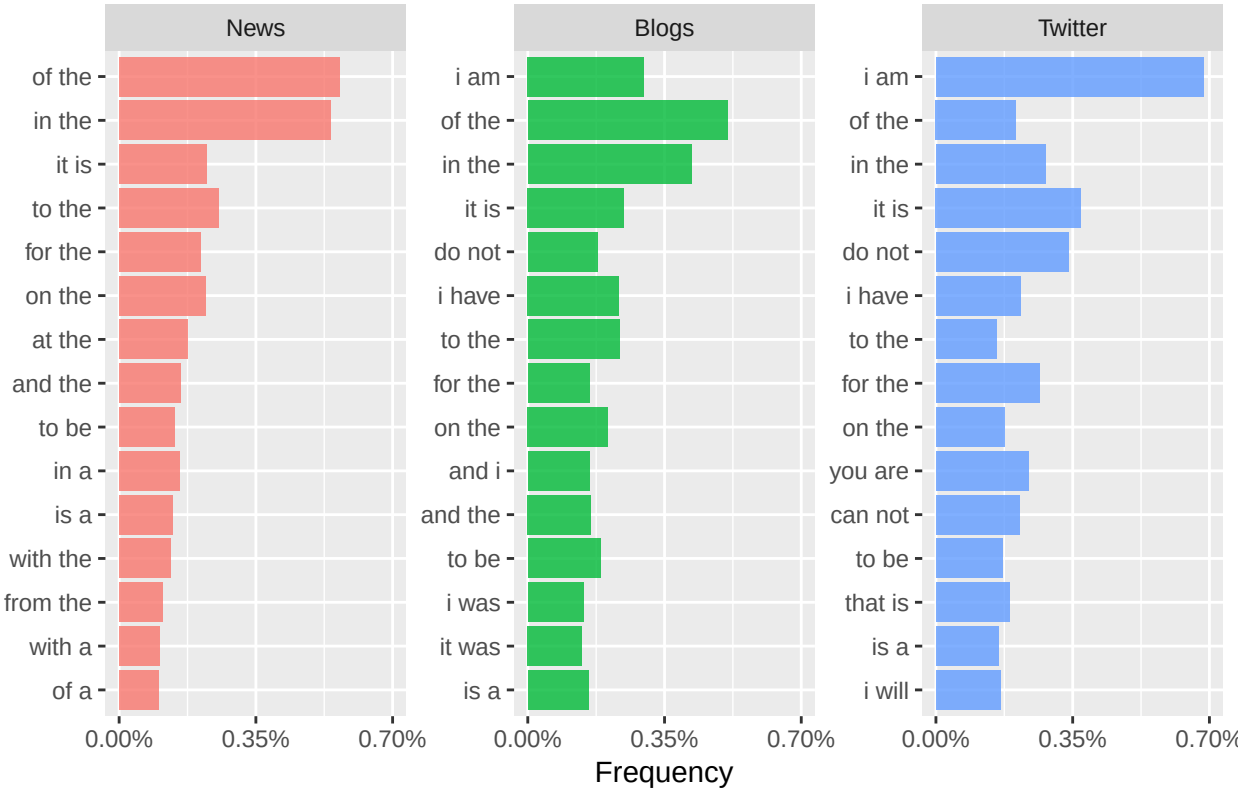
Table 2 shows some statistics for the unigrams, bigrams and trigrams of the training data. What is most interesting is that the source has a real impact on the number of unique n-grams. As a source becomes more informal (News->Blogs->Twitter) the rate at which the number of unique n-grams increases slows as n grows larger. This indicates that phrases used in more informal settings are used more often. Graphs of the top 15 n-grams by source are presented. It can be seen that the highest occurring phrases in Twitter feed occur far more often that the most common n-grams in the other sources. ("i am", and "i do not")

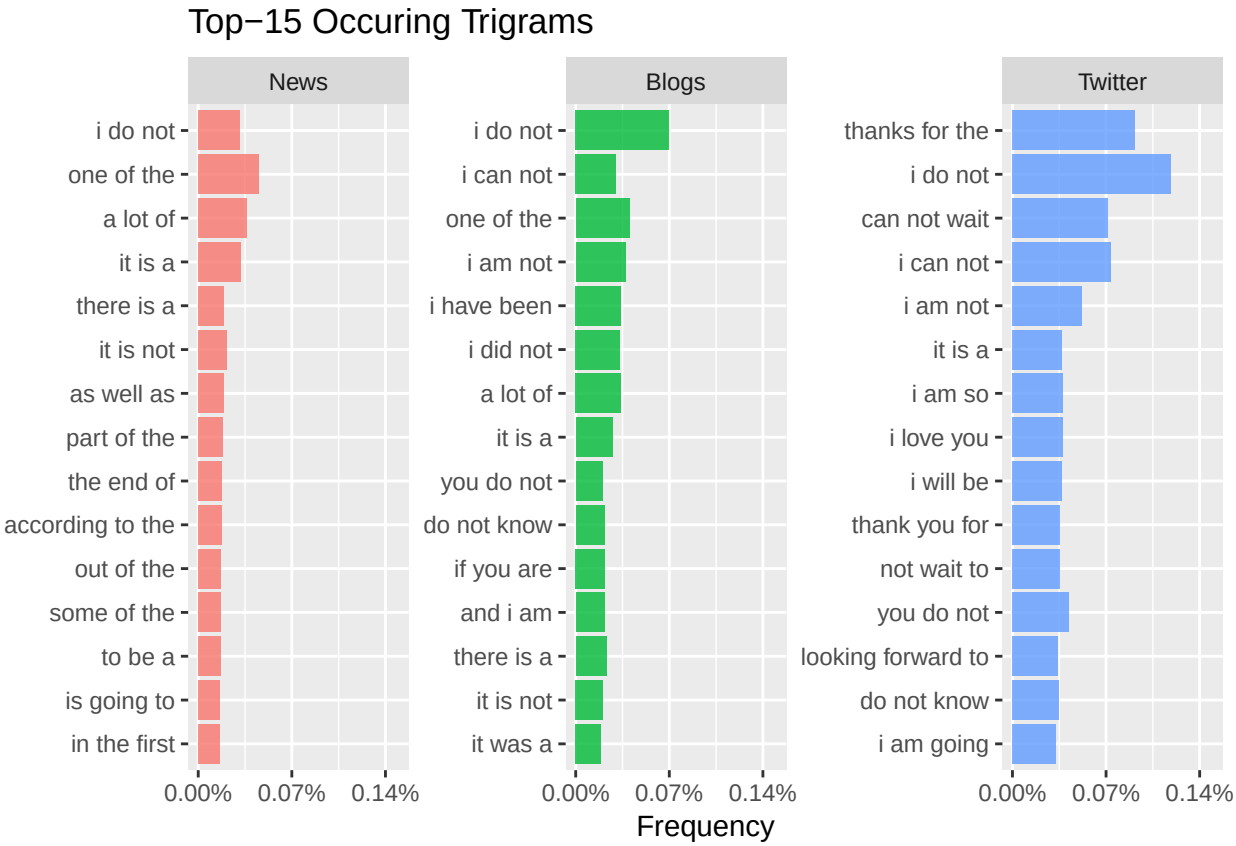
Also worth noting is the words that are found most frequently are *stop words*. In the Natural Language Processing (NLP) literature, *stop words* are normally stripped out of the corpus because they carry little meaning to the reader (or listener). However, for our purposes these represent important words for predicting what word might come next in a phrase. Therefore, no effort was made to eliminate these words from the corpus.

Table 2: N-Gram Analyis

		Unigrams		Bigrams		Trigrams	
Source	Excerpts	Tokens	N-Grams	Tokens	N-Grams	Tokens	N-Grams
News	606,146	20,526,430	195,957	19,920,788	4,207,150	19,319,047	11,449,016
Blogs	539,573	22,566,879	229,758	22,027,960	4,230,477	21,499,962	12,048,911
Twitter	1,416,089	18,110,744	210,229	16,694,728	3,242,147	15,279,336	8,210,259


Top−15 Occuring Unigrams
News
Blogs
Twitter
the
to
and
a
of
in
is
that
for
it
not
on
with
he
said
the
i
to
and
a
of
in
is
you
that
for
it
not
was
with
the
i
to
and
a
of
in
is
you
that
for
it
not
my
on
0%
3%
6%
Frequency


Top−15 Occuring Bigrams
News
Blogs
Twitter
of the
in the
it is
to the
for the
on the
at the
and the
to be
in a
is a
with the
from the
with a
of a
i am
of the
in the
it is
do not
i have
to the
for the
on the
and i
and the
to be
i was
it was
is a
i am
of the
in the
it is
do not
i have
to the
for the
on the
you are
can not
to be
that is
is a
i will
0.00%
0.35%
0.70%
Frequency



Next Steps

A survey of the NLP literature has indicated that a back off or smoothing algorithm will be necessary for a respectable prediction algorithm. Kneser-Ney smoothing seems to be the most preferable and therefore will be the first model implemented. Investigations have are also being conducted to find the best data structures to store and access the n-gram frequency data.

Appendix

All code used for the report is below. The Rmd and code are also available on GitHub.

A.1 Corpus Cleaning Function

```
clean_text <- function(text) {
  ## Convert encoding to ascii and normallize apostrophies
  Encoding(text) <- "latin1"
  text <- enc2utf8(text)
  text <- gsub("â€|â€˜", "'", text)
  text <- gsub("â€¦", "", text)
  text <- gsub("â€", "-", text)
  text <- iconv(text, "UTF-8", "ASCII", sub = "")

  ## Remove URLs & Hashtags & retweets
```

```
  text <- gsub("(ftp|http|www\\.|#)\\S*","", tolower(text))
  text <- gsub("(\\b)rt(\\b)", "", text)
  ## Replace _/- with space
  text <- gsub("[_-]", " ", text)
  ## Remove numbers
  text <- gsub("((\\d{1,3})(,\\d{3})*)|(\\d+))(.\\d+)?", "", text)

  ## Replace contractions
  text <- gsub("(\\b)(are|could|did|does|do|had|has|have|might|must|should|was|were|would)n'?t(\\b)", "
  text <- gsub("(\\b)can'?t|cannot(\\b)", "\\1can not\\2", text)
  text <- gsub("(\\b)isn'?t|ain'?t(\\b)", "\\1is not\\2", text)
  text <- gsub("(\\b)won't", "\\1will not", text)
  text <- gsub("(\\b)(how|i|should|they|we|must|what|who|would|you)'?ve(\\b)", "\\1\\2 have\\3", text)
  text <- gsub("(\\b)(it|they|what|who|you)'?ll(\\b)", "\\1\\2 will\\3", text)
  text <- gsub("(\\b)(he|i|she|we)'ll(\\b)", "\\1\\2 will\\3", text)
  text <- gsub("(\\b)(they|what|who|why|you)'?re(\\b)", "\\1\\2 are\\3", text)
  text <- gsub("(\\b)we're", "\\1we are", text)
  text <- gsub("(\\b)i'?m(\\b)", "\\1i am\\3", text)
  text <- gsub("(\\b)i'?d(\\b)", "\\1i would\\3", text)
  text <- gsub("(\\b)let'?s", "\\1let us", text)
  text <- gsub("(\\b)y'?all|ya'll", "\\1you all", text)
  text <- gsub("(\\b)(he|she|how|that|there|what|when|where|who|why)'?s(\\b)", "\\1\\2 is\\3", text)
  text <- gsub("(\\b)it's|'tis", "\\1it is", text)
  text <- gsub("(\\b)(how|what|where|who|why)'?d(\\b)", "\\1\\2 did\\3", text)
  ## Replace abreveations
  text <- gsub("(\\b)r(\\b)", "\\1are\\2", text)
  text <- gsub("(\\b)u(\\b)", "\\1you\\2", text)
  text <- gsub("(\\b)b4(\\b)", "\\1before\\2", text)
  text <- gsub("(\\b)b/?c(\\b)", "\\1because\\2", text)
  text <- gsub("(\\b)b(\\b)", "\\1be\\2", text)
  text <- gsub("(\\b)1st(\\b)", "\\1first\\2", text)
  text <- gsub("(\\b)2nd(\\b)", "\\1second\\2", text)
  text <- gsub("(\\b)3rd(\\b)", "\\1third\\2", text)
  text <- gsub("(\\w)'n(\\b)", "\\1ing\\2", text)
  text <- gsub("(\\b)n(\\b)", "\\1and\\2", text)
  text <- gsub("(\\b)w/o(\\b)", "\\1without\\2", text)
  text <- gsub("(\\b)w(\\b)", "\\1with\\2", text)
  text <- gsub("(\\b)w/(.)", "\\1with \\2", text)
  text <- gsub("(\\s+)@(\\s+)", "\\1at\\2", text)
  text <- gsub("(\\s+)&(\\s+)", "\\1and\\2", text)

  ##replace repeating characters
  text <- gsub("([a-z])\\1{2,}", "\\1", text)
}
```

A.2 Load Corpus and Analyze

```
maxlines <- -1

## Load Data
twtrtext <- read_lines("data/en_US/en_US.twitter.txt", n_max = maxlines)
blogtext <- read_lines("data/en_US/en_US.blogs.txt", n_max = maxlines)
```

```
newstext <- read_lines("data/en_US/en_US.news.txt", n_max = maxlines)
twtrn <- length(twtrtext); blogn <- length(blogtext); newsn <- length(newstext)

## Create Corpus
corpus <- rbind(
  data_frame(source = as.factor(rep("News", newsn)), excerpt = 1:newsn,
             group = excerpt %% 7, text = newstext),
  data_frame(source = as.factor(rep("Blogs", blogn)), excerpt = 1:blogn,
             group = excerpt %% 7, text = blogtext),
  data_frame(source = as.factor(rep("Twitter", twtrn)), excerpt = 1:twtrn,
             group = excerpt %% 7, text = twtrtext))

## Generate ugrams on raw text for analysis
corpusugramraw <- unnest_tokens(corpus, ugram, text) %>% group_by(source) %>% count(ugram)

## Clean and normalize corpus
cluster <- makeCluster(7, type = "FORK")
registerDoParallel(cluster)
corpus <- foreach(i=0:6, .combine = rbind) %dopar%
  mutate(filter(corpus, group == i), text = clean_text(text))
stopCluster(cluster)

## Generate ugrams on cleaned text for analysis
corpusugram <- unnest_tokens(corpus, ugram, text) %>% group_by(source) %>% count(ugram)

## Generate stats
corpastats <- list(summarise(group_by(corpus, source), ln = n()),
                   summarise(corpusugramraw, rtok = sum(n), rugram = n()),
                   summarise(corpusugram, ctok = sum(n), cugram = n())) %>%
  Reduce(function(dtf1, dtf2) full_join(dtf1, dtf2, by = "source"), .)

kable(corpastats, format = "latex", booktabs = "T", format.args = list(big.mark = ","),
      caption = "Overview of Corpus",
      col.names = c("Source", "Excerpts", "Tokens", "Unigrams", "Tokens", "Unigrams"))  %>%
  add_header_above(c(" " = 2, "Raw" = 2, "Cleaned" = 2)) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

A.3 Sampling and N-Gram Generatiom

```
set.seed(1234)
inTrain <- createDataPartition(corpus$source, p=0.60, list=FALSE)

## Generate unigrams
if (!file.exists("data/ugrams.rds")) {
  ugrams <- unnest_tokens(corpus[inTrain,], ugram, text, token = "ngrams", n = 1) %>%
    group_by(source) %>% count(ugram)
  ugramstats <- summarise(ugrams, n_toks = sum(n), n_grams = n())
  ugrams <- left_join(ugrams, ugramstats) %>% mutate(freq = n/n_toks)
  saveRDS(ugrams, "data/ugrams.rds")
} else {
  ugrams <- readRDS("data/ugrams.rds")
  ugramstats <- summarise(ugrams, n_toks = sum(n), n_grams = n())
```

```
}

## Generate bigrams
if (!file.exists("data/bgrams.rds")) {
  bgrams <- unnest_tokens(corpus[inTrain,], bgram, text, token = "ngrams", n = 2) %>%
    group_by(source) %>% count(bgram)
  bgramstats <- summarise(bgrams, n_toks = sum(n), n_grams = n())
  bgrams <- left_join(bgrams, bgramstats) %>% mutate(freq = n/n_toks)
  saveRDS(bgrams, "data/bgrams.rds")
} else {
  bgrams <- readRDS("data/bgrams.rds")
  bgramstats <- summarise(bgrams, n_toks = sum(n), n_grams = n())
}

## Generate trigrams
if (!file.exists("data/tgrams.rds")) {
  tgrams <- unnest_tokens(corpus[inTrain,], tgram, text, token = "ngrams", n = 3) %>%
    group_by(source) %>% count(tgram)
  tgramstats <- summarise(tgrams, n_toks = sum(n), n_grams = n())
  tgrams <- left_join(tgrams, tgramstats) %>% mutate(freq = n/n_toks)
  saveRDS(tgrams, "data/tgrams.rds")
} else {
  tgrams <- readRDS("data/tgrams.rds")
  tgramstats <- summarise(tgrams, n_toks = sum(n), n_grams = n())
}

corpastats <- list(summarise(group_by(corpus[inTrain,], source), ln = n()),
                   ugramstats, bgramstats, tgramstats) %>%
  Reduce(function(dtf1, dtf2) full_join(dtf1, dtf2, by = "source"), .)

kable(corpastats, format = "latex", booktabs = "T", format.args = list(big.mark = ","),
      caption = "N-Gram Analysis",
      col.names = c("Source", "Excerpts", rep(c("Tokens", "N-Grams"), 3)))  %>%
  add_header_above(c(" " = 2, "Unigrams" = 2, "Bigrams" = 2, "Trigrams" = 2)) %>%
  kable_styling(latex_options = c("striped", "hold_position"))

top_n(ugrams, 15) %>%
  ggplot(aes(reorder(ugram, n), freq, fill = source)) +
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_wrap(~ source, ncol = 3, scales = "free_y") +
  coord_flip() +
  scale_y_continuous(limits=c(0.0, 0.06), breaks=c(0, 0.03, 0.06), labels=scales::percent)+
  labs(title = "Top-15 Occuring Unigrams", x = NULL, y = "Frequency")

top_n(bgrams, 15) %>%
  ggplot(aes(reorder(bgram, n), freq, fill = source)) +
  geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
  facet_wrap(~ source, ncol = 3, scales = "free_y") +
  coord_flip() +
  scale_y_continuous(limits=c(0.0, 0.007), breaks=c(0, 0.0035, 0.007), labels=scales::percent)+
  labs(title = "Top-15 Occuring Bigrams", x = NULL, y = "Frequency")

top_n(tgrams, 15) %>%
```

```r
ggplot(aes(reorder(tgram, n), freq, fill = source)) +
geom_bar(alpha = 0.8, stat = "identity", show.legend = FALSE) +
facet_wrap(~ source, ncol = 3, scales = "free_y") +
coord_flip() +
scale_y_continuous(limits=c(0.0, 0.0015), breaks=c(0, 0.0007, 0.0014), labels=scales::percent)+
labs(title = "Top-15 Occuring Trigrams", x = NULL, y = "Frequency")
```