Capstone Milestone Report

Rhowell

2022-11-29

Summary

This is the Milestone Report for the John Hopkins University's Data Science Specialization Capstone offered by Coursera. The Capstone consists of creating a predictive text generator based off of corpa of text from SwiftKey. This Milestone Report will read the corpa, preform basic explanatory data analysis, and finally describe several ways forward for the final predictive text model.

Load the Data

```r
train_url <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
train_data_file <- "data/Coursera-SwiftKey.zip"

if (!file.exists('data')) {
    dir.create('data')
}
if (!file.exists("data/final/en_US")) {
    tempFile <- tempfile()
    download.file(train_url, tempFile)
    unzip(tempFile, exdir = "data")
    unlink(tempFile)
}

# blogs
blogs_file_name <- "data/final/en_US/en_US.blogs.txt"
con <- file(blogs_file_name, open = "r")
blogs <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)

# news
news_file_name <- "data/final/en_US/en_US.news.txt"
con <- file(news_file_name, open = "r")
news <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)

# twitter
twitter_file_name <- "data/final/en_US/en_US.twitter.txt"
con <- file(twitter_file_name, open = "r")
twitter <- readLines(con, encoding = "UTF-8", skipNul = TRUE)
close(con)
```

Table 1:

File	FileSize	Lines	Characters	Words	WPL.Min	WPL.Median	WPL.Max
en_US.blogs.txt	200 MB	899288	206824505	37570839	0	28	6726
en_US.news.txt	196 MB	1010242	203223159	34494539	1	32	1796
en_US.twitter.txt	159 MB	2360148	162096241	30451170	1	12	47

```
rm(con)
```

Exploratory Data Analysis

Initially the corpa are in three files, one each for Twitter, news, and blogs. I will combine them into a single corpus as well as tidy some of the text.

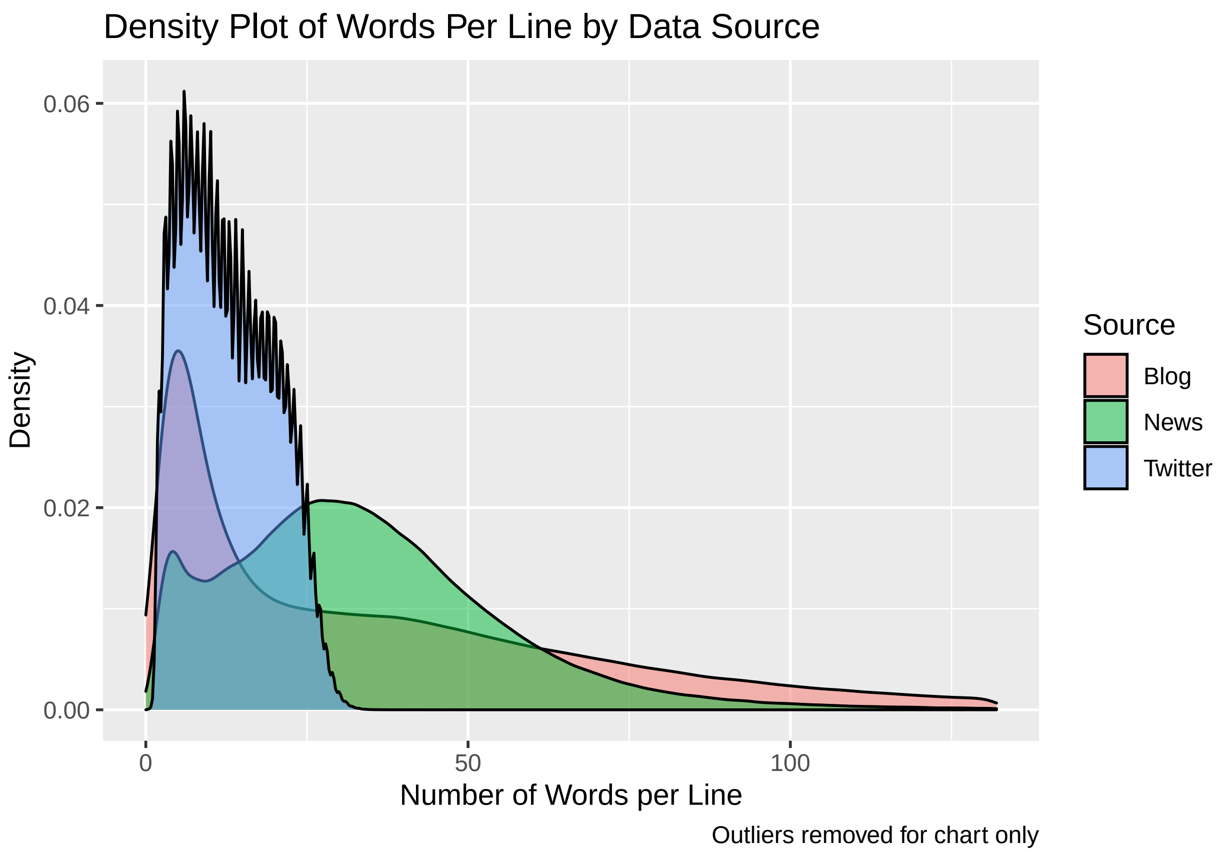
Basic Exploratory Summary

Let us first explore the three separate files to find basic summaries of the three files, including word counts, line counts and basic data tables.

From the table, whereas the files appear about the same size, the files differ in number or records (lines) and average words per line. Twitter has the most records (lines), but the fewest words per line. On the other hand, blogs have the fewest records (lines), but the most words per line. News falls in the middle of both records (lines) and words per line. Intuitively, this makes sense since Twitter caps the characters per tweet. Additionally, blogs tend to be smaller than news articles due to the journalistic style of blogs, versus the informational style news articles.

Boxplot of Words Per Line

```
library(reshape2)
library(dplyr)
library(ggplot2)

long_wpl <- melt(wpl, id="x") %>%
  mutate(L1 = as.factor(L1))

ggplot(data = long_wpl)+
  geom_density(aes(x = value, fill = L1), alpha = 0.5, outlier.shape = NA) +
  scale_fill_discrete(name = "Source", labels=c('Blog', 'News', 'Twitter')) +
  scale_x_continuous(limits = quantile(long_wpl$value, c(0, 0.99))) +
  labs(title = "Density Plot of Words Per Line by Data Source", caption = "Outliers removed for chart or
  xlab("Number of Words per Line") +
  ylab("Density")
```


Density Plot of Words Per Line by Data Source

Outliers removed for chart only

The boxplot (with outliers removed) depicts a slightly different story of the data. None are normally distributed, but all are right-skewed, as predicted by Benford's Law.

Data Preparation

The data sources will be sampled at 1% to train the predictive model more effectively.

```
# set seed
set.seed(424242)

# sample the sources
sample_blogs <- sample(blogs, length(blogs) * 0.01, replace = FALSE)
sample_news <- sample(news, length(news) * 0.01, replace = FALSE)
sample_twitter <- sample(twitter, length(twitter) * 0.01, replace = FALSE)

# remove all non-English characters from the sampled data
sample_blogs <- iconv(sample_blogs, "latin1", "ASCII", sub = "")
sample_news <- iconv(sample_news, "latin1", "ASCII", sub = "")
sample_twitter <- iconv(sample_twitter, "latin1", "ASCII", sub = "")

# merge into one dataframe
sample_data <- c(sample_blogs, sample_news, sample_twitter)
```

The data will now be cleaned and merged into a single corpus for use by the text predictor model later.

Table 2: First 10 Documents

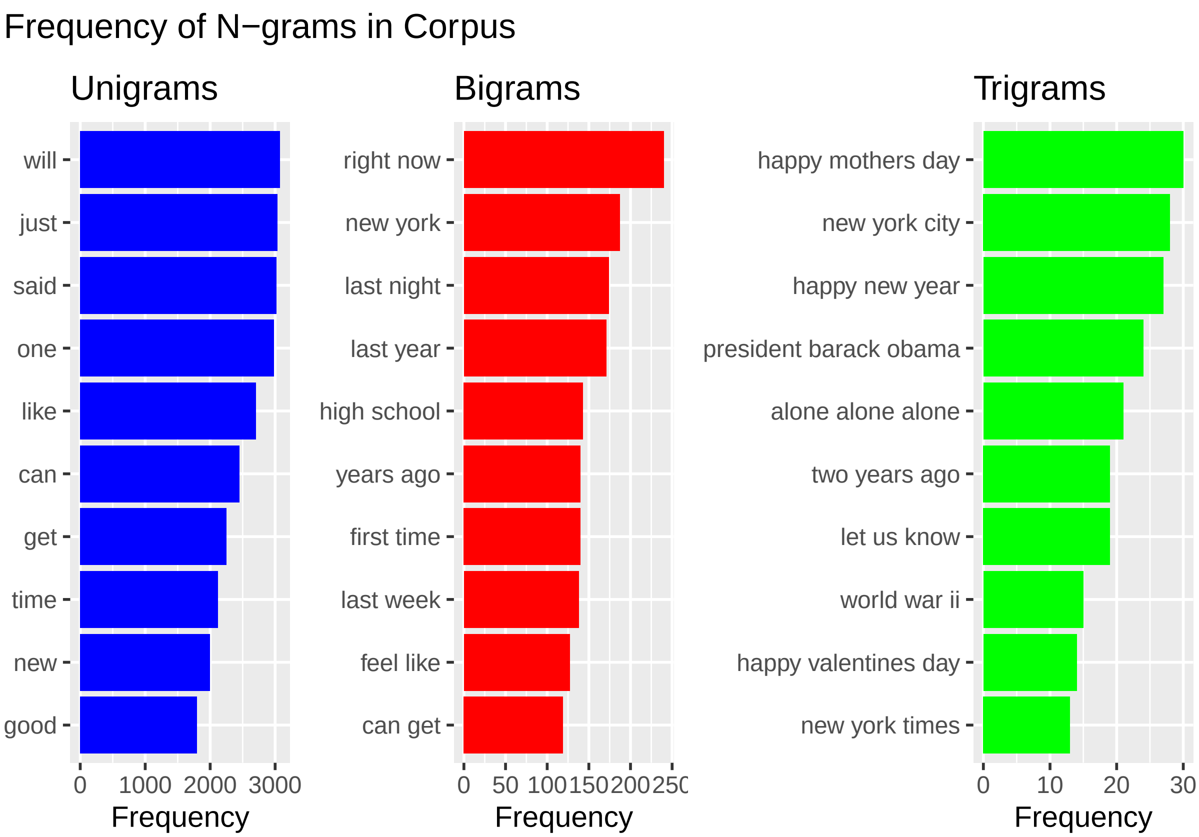
allergic plums break rash face eat stop eating
thanks follow dawn
thank cari wish many great weeks
best work years grey treasure
product engineering session missed
happy birthday babyy changed lifes much proud u congrats
indeed sooner wife can land interview dartmouth
know nagra sts former landlord vashon prop master wonder years much cool junk
thingsthatannoyme last one going sleep clean face waking big red pimple
know gonna haut gonna need pic u totheter u know u one u sent u g wheni

```
library(tm)
library(sentimentr)

corpus <- VCorpus(VectorSource(sample_data))  # Build corpus
toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))  # Space converter function
corpus <- tm_map(corpus, toSpace, "(f|ht)tp(s?)://(.*)[.][a-z]+")  # Mutate URLs to spaces
corpus <- tm_map(corpus, toSpace, "@[^\\s]+")  # Mutate Twitter handles to spaces
corpus <- tm_map(corpus, toSpace, "\\b[A-Z a-z 0-9._ - ]*[@](.*?)[.]{1,3} \\b")  # Mutate email pattern
corpus <- tm_map(corpus, tolower)  # Convert all words to lowercase
corpus <- tm_map(corpus, removeWords, stopwords("english"))  # Remove common English stop words
corpus <- tm_map(corpus, removePunctuation)  # Remove punctuation marks
corpus <- tm_map(corpus, removeNumbers)  # Remove numbers
corpus <- tm_map(corpus, stripWhitespace)  # Trim whitespace
corpus <- tm_map(corpus, removeWords, lexicon::profanity_arr_bad)  # Remove profanity
corpus <- tm_map(corpus, PlainTextDocument)  # Convert to plain text documents

# Display first 10 observations from single corpus
corpus_text <- data.frame(text = unlist(sapply(corpus, '[', "content")), stringsAsFactors = FALSE)
kable(tail(corpus_text$text, 10),
      row.names = FALSE,
      col.names = NULL,
      align = c("l"),
      caption = "First 10 Documents") %>%
        kable_styling(position = "left")
```

Data Features of Tokenization by N-gram

Next in the exploratory data analysis is looking at tokenization by n-gram.

```
library(dplyr)
library(tidyr)
library(tidytext)
library(patchwork)

corpus_text_tibble <- tibble(line = 1:nrow(corpus_text), text = corpus_text$text)

corpus_text_unigram <- corpus_text_tibble %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot() +
  geom_col(aes(n, reorder(word, n)), fill = "blue") +
  labs(title = "Unigrams") +
  ylab(NULL) +
  xlab("Frequency")

corpus_text_bigram <- corpus_text_tibble %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  filter(!is.na(bigram)) %>%
  count(bigram, sort = TRUE) %>%
  top_n(10) %>%
  ggplot() +
  geom_col(aes(n, reorder(bigram, n)), fill = "red") +
  labs(title = "Bigrams") +
  ylab(NULL) +
  xlab("Frequency")

corpus_text_trigram <- corpus_text_tibble %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  filter(!is.na(trigram)) %>%
  count(trigram, sort = TRUE) %>%
  top_n(10) %>%
  ggplot() +
  geom_col(aes(n, reorder(trigram, n)), fill = "green") +
  labs(title = "Trigrams") +
  ylab(NULL) +
  xlab("Frequency")

corpus_text_unigram +
  corpus_text_bigram +
  corpus_text_trigram +
  plot_annotation(
  title = "Frequency of N-grams in Corpus"
)
```



Way Forward

From here, the data will be used to create a predictive text generator based off of the n-grams above. It will be deployed on Shinyapps.io, and will be interactive based on user input. There are several possible ways to use the model. Most likely will be to search the trigram for prediction, and use the next word if the phrase is found. If it is not found, second search the bigram for prediction, and use the next word if the phrase is found. Finally, if the phrase is not found in the bigram model, using the unigram for the next word prediction.