Text Prediction App - Milestone Report

DMK

8/24/2020

### Summary

This final project (Capstone) for Data Science Specialization requires developing a text prediction Shiny App. The app is “taught” to predict next word in a sentence by training it using three files provided via Coursera / Swiftkey. For the purpose of this submission, English language files are utilized.

### Steps

It is evident that the files are large - so, as first step I’ll evaluate the number of lines of text in each without actually reading them into R. This will help determine a number of lines that should be used as a good representative set for the app to “learn” from.

#### Setup

* Load required packages

### Explore Files

* Gather information on total lines & average word count per line in datasets

Twitter <- file("./final/en\_US/en\_US.twitter.txt")  
Blogs <- file("./final/en\_US/en\_US.blogs.txt")  
News <- file("./final/en\_US/en\_US.news.txt")  
  
length(readLines(Twitter, skipNul = TRUE))

## [1] 2360148

wordcount(readLines(Twitter, skipNul = TRUE), count.function = mean)

## [1] 12.86936

length(readLines(Blogs, skipNul = TRUE))

## [1] 899288

wordcount(readLines(Blogs, skipNul = TRUE), count.function = mean)

## [1] 41.51521

length(readLines(News, skipNul = TRUE))

## [1] 1010242

wordcount(readLines(News, skipNul = TRUE), count.function = mean)

## [1] 34.02406

### Collect Sample Corpus

Approx. 2% of data is used in training set

set.seed(5432)  
TwitSample <- sample(readLines(Twitter, skipNul = TRUE), size = 50000, replace = TRUE)  
BlogSample <- sample(readLines(Blogs, skipNul = TRUE), size = 21500, replace = TRUE)  
NewsSample <- sample(readLines(News, skipNul = TRUE), size = 19000, replace = TRUE)  
TrainSet <- c(TwitSample, BlogSample, NewsSample)  
rm(TwitSample, BlogSample, NewsSample)  
# TrainSet <- Corpus(VectorSource(Train))

### Cleanse Data

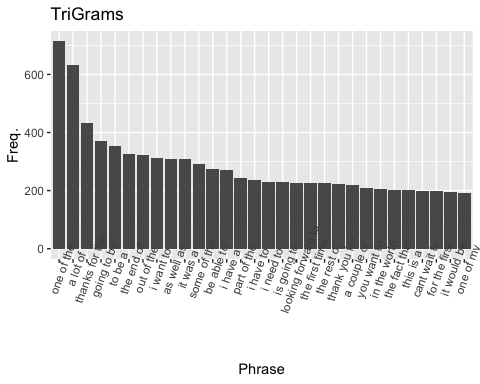
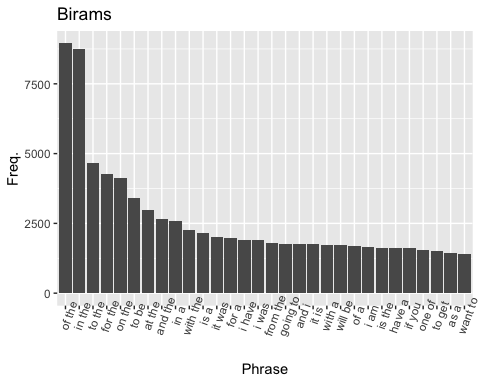
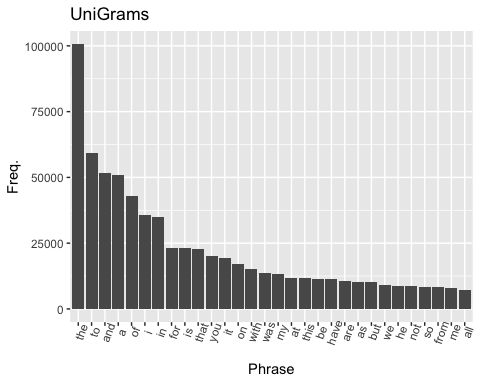
* Data is cleansed by removing punctuations, white spaces, numbers, etc.

TrainSet <- removeWords(TrainSet, Profanities)  
TrainSet <- tolower(TrainSet)  
TrainSet <- removePunctuation(TrainSet)  
TrainSet <- removeNumbers(TrainSet)  
TrainSet <- str\_squish(TrainSet)  
TrainSet <- str\_trim(TrainSet)

* After cleansing data, 1-, 2-, & 3-grams are created and saved as dataframes

#UniGram  
TrainSet <- TrainSet[str\_count(TrainSet, "\\s+")>1]  
Gram1 <- ngram(TrainSet, n=1)  
UniGram <- get.phrasetable(Gram1)  
  
#BiGram  
TrainSet <- TrainSet[str\_count(TrainSet, "\\s+")>2]  
Gram2 <- ngram(TrainSet, n=2)  
BiGram <- get.phrasetable(Gram2)  
  
#TriGram  
TrainSet <- TrainSet[str\_count(TrainSet, "\\s+")>3]  
Gram3 <- ngram(TrainSet, n=3)  
TriGram <- get.phrasetable(Gram3)

### Words / phrases appearing most frequently in each n-gram are:



### Next Steps

* Utilize exported Uni-, Bi-, and Tri-grams to build a Shiny App that predicts / proposes next word based on text entered.