Capstone Project for Data Science Specialization

Exploratory Data Anaysis Report

Jay Yanamandala

11-29-2021

Contents

https://rpubs.com/jyanamandala/Milestone_Report ## 1. Executive Summary The goal of this report is to mainly provide a brief update on the exploratory analysis done to achieve the goal towards an eventual app on Shiny, and certification as a Data Science Specialist awarded by John Hopkin's University

The corpus data is provided by SwiftKey's Natural Language Processing (NLP) project, will be processed to create N-grams Given the size of text files provided for three corpus, and available machine architecture, project will be limited to creating bigrams, trigrams, and quadgrams to make predictions of next word that will most likely be typed by User.

2. Main flow steps involved in the Capstone Project

2.1 Downloading data

The training data for this Capstone SwiftKey project is downloaded from Coursera Site. Only en_US* files are unzipped into 'data' directory for NLP analysis.

```
## Load libraries needed
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(dplyr))

## Code to download
source("scripts/extFiles.R")
source("scripts/processLines.R")
source("scripts/getFileInfo.R")

# download_data("en_US", "original")
zipFile <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
extFiles(zpath=zipFile, datadir="data", lname="en_US")

## Preliminary Data Analysis and Setup for Processing
```

```
## Create tibbles and gather files information
news <- processLines(file = "data/news.txt", num = Inf)
news_df <- tibble(line=1:length(news), news)
names(news_df) <- c("line","word")
newsInfo<-getFileInfo(file = "data/news.txt", txtFile = news)

## Create blogs tibble
blogs <- processLines(file = "data/blogs.txt", num = Inf)
blogs_df <- tibble(line=1:length(blogs), blogs)
names(blogs_df) <- c("line","word")
blogsInfo<-getFileInfo(file = "data/blogs.txt", txtFile = blogs)

## Create twitter tibble
twitter <- processLines(file = "data/twitter.txt", num = Inf)
twitter_df <- tibble(line=1:length(twitter), twitter)
names(twitter_df) <- c("line","word")
twitterInfo<-getFileInfo(file = "data/twitter.txt", txtFile = twitter)
```

2.2 Gathering Datasets stats

```
### Information about the text files
## Create data.frame with file size, length, longest line info
`File Size` <- c(newsInfo$FileSize,blogsInfo$FileSize,twitterInfo$FileSize)
`File Length` <- c(newsInfo$FileLength,blogsInfo$FileLength,twitterInfo$FileLength)
`Longest Line` <- c(newsInfo$LongestLineLength,blogsInfo$LongestLineLength,twitterInfo$LongestLineLengt

## Combine files into a dataframe
all_files_info <- data.frame(`File Size`, `File Length`, `Longest Line`)
row.names(all_files_info) <- c("News", "Blogs", "Twitter")
print(all_files_info)

## Remove unneeded VARs from memory
rm(news, newsInfo,twitter,blogs,twitterInfo, blogsInfo)
```

2.3 Preprocessing data to create tibbles

This preprocess step includes **cleaning** (removing profane words, numbers, puncutations, spaces, etc), *tokenization* ***to create n-grams,*** *frequency* ***capture of words for prediction,*** *merging* of data frames (needed for Shiny App).
The following are the steps incolved in preprocessing of the text files
1. Convert capital letters to lower case
2. Remove Numbers
3. Remove White spaces
4. Remove Emojis
5. Remove AlphaNumerals
6. Remove Profanity words

2.4 Creating N-Grams

For creating n-grams, We will be looking at two different approaches:
1. Using tidy

2. Using tm

Note: Creating trigrams and quadgrams is very memory intensive, especially for blogs.txt. If it is really needed to process huge files, best to split the dataframe into manageable chuncks -or- skip reading lines if greater than a certain length.

Since final step in the project is to create a Shiny App, only 33% of blogs.txt and twitter.txt will be used in this project to showcase Shiny App.

2.4.1 Using tidy unnest tokens 'tidyverse' package Citation: Text Mining with R a book by Julia Silge & David Robinson The Life-Changing Magic of Tidying Text | Julia Silge

Here are is sample tidy code for creating unigram:

```
data(stop_words)
df_sort <- df %>%
    unnest_tokens(word, word) %>%
    anti_join(stop_words) %>%
    count(word, sort=T)
```

2.4.2 Using TextMatrixDocumentation 'tm' package Citation: Jaehyeon's Blog Part1, Part 2, and Part 3

```
require(parallel); require(tm)
n_cores <- detectCores() -1
cl <- makeCluster(n_cores, type="PSOCK") ## on windows
invisible(clusterEvalQ(cl, library(tm)))
clusterExport(cl,"vdocs", envir = environment()) ## Export variable and cluster
parLapply(cl, 1, function(x)
{
  vdocs <- tm_map(vdocs, content_transformer(tolower))        ## to lowercase
  vdocs <- tm_map(vdocs, stripWhitespace)   ## strip white space
  ## You can add more tm_map steps
}
stopCluster(cl)
```

2.4.3 Merge each Corpus ngram to one file After creating ngrams (bigram, trigram, quadgram) Document matrix for each corpus, the following steps will be performed:
1. Merge the 3 ngram dataframes into one
2. Reduce them by 'full_join' based on 'word' column
3. Transform the merged dataframe and create 'total' column
4. Separate the words, and drop the count for individual ngram
5. Merge the final dataframe for each corpus into one
6. Efficient storage of the n-gram model (Markov Chains)

2.4.4 Prediction

1. For the model to be more efficient, depending on the memory requirement for Shiny App on server, may have to drop some rows based on frequency

2. Use backoff models to estimate probability of unobserved n-grams

3. Create train and test data to build models for prediction algorithm to predict next word

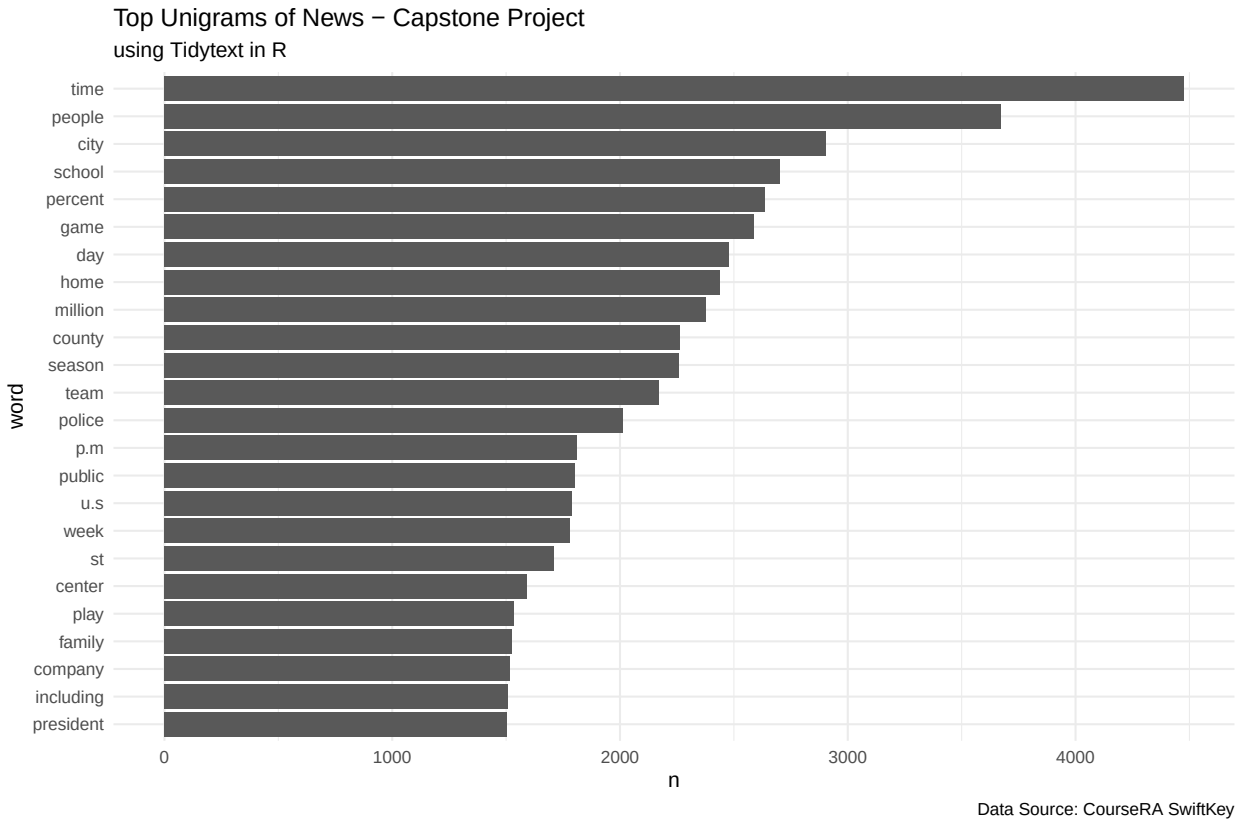
3. Dataset statistics

3.1 Summary Statistics of Dataset

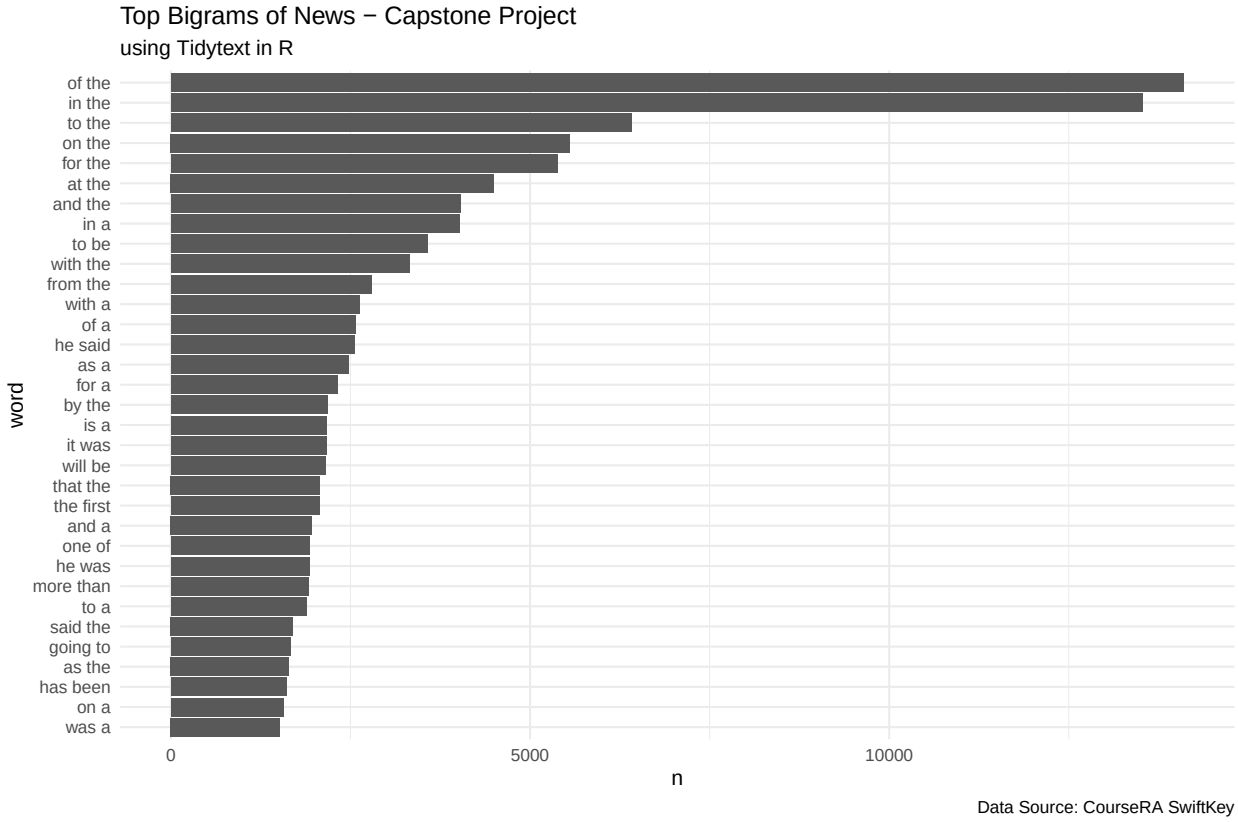
Stats	News	Blogs	Twitter
File.Size	196.2775	200.4242	159.3641
File.Length	77259	899288	2360148
Longest.Line	5760	40833	140

4. N-Gram plots

4.1 unigram plot



4.2 bigram plot



5. Recap and Next steps SwiftKey Capstone Project

1. For single term words, the words time, people and day are very common. We also see the word 'rt' which might stand for retweet from twitter, and should be added to stop words. We also see teh words game, team, and night which all points towards a sports game. Then there's the word 'lol' which I believe comes from twitter as well.

2. For bigrams, we see common spoken English words, other Corpus documents, blogs and twitter will show different terms snce I plan to use **anti_join(stop_words)**. Assumption is this should give us better prediction of next word.

3. This analysis helped me understand many different methods to explore, analyze, and analyze the various nuances in datascience data structure and manipulation. In addition learned to execute **tm_map** and **TermDocumentMatrix** in parallel. Alos understood the limitation of machines, and managing memory/cpu.

4. After creating n-grams for individual corpus, next step is to build a compact model that combines all three corpus and split the model into train, validation, and test sets and chose best performing model to predict next word

5. Apply all the methods to build a Shiny App. Users can type words in a text box, and the app will predict next set of words for User to choose from.

6. Final step is to build a slide deck to present to Users on how to use the product