Coursera Data Science Capstone: Milestone Report

Manojkumar Parmar

2/12/2017

Contents

1 Introduction **2**

2 Procesing Data **2**
2.1 Fetching . . . 2
2.2 Summary Statistics . . . 2
2.3 Cleaning Data . . . 3
2.4 Tokenization of data . . . 4

3 Visualization **5**
3.1 Histogram . . . 5
3.1.1 Function . . . 5
3.1.2 Visualization . . . 7
3.1.2.1 Top 10 UniGrams across all datasets . . . 7
3.1.2.2 Top 10 Bigrams across all datasets . . . 7
3.1.2.3 Top 10 TriGrams across all datasets . . . 8
3.1.2.4 Top 10 QuadGrams across all datasets . . . 9
3.1.3 Discussion . . . 10
3.2 Unique Word Analysis for 50% and 90% coverage . . . 10
3.2.1 Function . . . 11
3.2.2 Visualization . . . 12
3.2.2.1 Coverage of UniGrams across all datasets . . . 12
3.2.2.2 Coverage of Bigrams across all datasets . . . 12
3.2.2.3 Coverage of TriGrams across all datasets . . . 13
3.2.2.4 Coverage of QuadGrams across all datasets . . . 14
3.2.3 Discussion . . . 15
3.3 WordCloud . . . 15
3.3.1 Function . . . 16
3.3.2 Visualization . . . 17
3.3.2.1 WordCloud of UniGrams across all datasets . . . 17
3.3.2.2 WordCloud of BiGrams across all datasets . . . 18
3.3.3 Discussion . . . 19

4 Conclusion **19**

5 Reproducibility **20**

Executive Summary

The goal of this project is to analyse the corpus data provided from various sources like tweeter, news sites & blog sites and to come up with a strategy for predicting next word from partial sentences/words.This strategy/modeling technique will be used to build shiny app. This reports includes data fetching, generating summary statistics on data and then cleaning of data using principles of natural language processing. On cleaned data, first ngrams up to quad grams are generated and then various exploratory techniques are used to provide insight of data & underlying pattern in data. Histogram (Quantitative technique) of frequently occurring words provides with top words and provides nature of distribution of top words. Unique word

coverage of 50% & 90% (Quantitative technique) provides a detail about how much words are necessary to cover up 50% & 90% of corpus. It also helps to understand underlying distribution of words & ngrams. Word-cloud (Visual technique) is used to represent most occurring terms and which is helpful in unearthing dependent patterns. Tweeter data stands out differently in various analysis and hence it is needed to be treated different in final modeling.

---

1 Introduction

The data provided in the course site comprises four sets of files (de_DE - Danish, en_US - English,fi_FI - Finnish an ru_RU - Russian), with each set containing 3 text files with texts from blogs, news/media sites and twitter. In this analysis we will focus English set of files: . en_US.blogs.txt . en_US.news.txt . en_US.twitter.txt. For analysis, English corpus is used.

This report aims to let the reader understand the training dataset used to develop the text prediction engine of a shiny application that is able to predict the next word based on the user's input. Some of the key tasks that will be described in this report include data acquisition, data pre-processing as well as data exploration in the field of natural language processing.

2 Procesing Data

2.1 Fetching

```r
if(! file.exists("./00_Data/All_Read.RData")){
        tweetFile <- file("./00_Data/final/en_US/en_US.twitter.txt", "r")
        blogFile <- file("./00_Data/final/en_US/en_US.blogs.txt","r")
        newsFile <- file("./00_Data/final/en_US/en_US.news.txt","r")
        contentTweet <- readLines(tweetFile)
        contentBlog <- readLines(blogFile)
        contentNews <- readLines(newsFile)
        close(tweetFile)
        close(blogFile)
        close(newsFile)
        save.image("./00_Data/All_Read.RData")
}else{
        load("./00_Data/All_Read.RData")

}
```

2.2 Summary Statistics

The following table provides an overview of the imported data. In addition to the size of each data set, the number of lines and words are displayed.

File Name	File Size in Megabyte	Line Count	Word Count
Blogs	200.42	899288	37334147
News	196.28	1010242	34372530
Twitter	159.36	2360147	30373539

Dataset is too big hence only 1% of tweet data, 1.5% of blog data and 1.5% of news data is used. Also combined version of all data is also prepared.

```
set.seed(1)
contentTweet <- sample(contentTweet, length(contentTweet)* 0.01)
contentBlog <- sample(contentBlog, length(contentBlog) * 0.015)
contentNews <- sample(contentNews, length(contentNews) * 0.015)
contentPart <-c(contentTweet, contentBlog, contentNews)
```

2.3 Cleaning Data

Cleaning Function comprising multiple elements

- Removal of Punctuation
- Removal of Numbers
- Removal of URLs & Email id
- Removal of Twitter tags & username
- Removal of profane words (DataSet1, DataSet2)
- Removal of English stop words
- Removal of White spaces
- Stemming of documents

```
tm_pre_process <- function(data) {
        library(tm)

        # Create patterns to elimina special code and other patterns
        # URLs
        urlPat <- function(x)
                gsub("(ftp|http)(s?)://.*\\b", "", x)
        # Emails
        emlPat <-
                function(x)
                        gsub("\\b[A-Z a-z 0-9._ - ]*[@](.*?)[.]{1,3} \\b", "", x)
        # Twitter tags
        tt <- function(x)
                gsub("RT |via", "", x)
        # Twitter Usernames
        tun <- function(x)
                gsub("[@][a - zA - Z0 - 9_]{1,15}", "", x)
        #Remove profane words
        pwdat <-
                read.table(
                        "./00_Data/final/profane_words/en_bws.txt",
                        header = FALSE,
                        sep = "\n",
                        strip.white = TRUE
                )
        names(pwdat) <- "Profane Words"
        pwdat1 <- read.csv("./00_Data/final/profane_words/Terms-to-Block.csv")
        pwdat1 <- pwdat1[-(1:3), 2]
        pwdat1 <- gsub(",","",pwdat1)
        stpWrdList <- c(as.character(pwdat[,1]),
                        pwdat1,
                        stopwords("english")
        )

        corpusTitle = Corpus(VectorSource(data))
        corpusTitle = tm_map(corpusTitle, tolower)
        corpusTitle = tm_map(corpusTitle, PlainTextDocument)
        corpusTitle = tm_map(corpusTitle, removePunctuation)
        corpusTitle = tm_map(corpusTitle, removeNumbers)
        corpusTitle = tm_map(corpusTitle, urlPat)
        corpusTitle = tm_map(corpusTitle, emlPat)
        corpusTitle = tm_map(corpusTitle, tt)
        corpusTitle = tm_map(corpusTitle, tun)
        corpusTitle = tm_map(corpusTitle, removeWords, stpWrdList)
        corpusTitle = tm_map(corpusTitle, stemDocument)
        corpusTitle = tm_map(corpusTitle, stripWhitespace)
        corpusTitle
}
```

Using above mentioned pre processing function, corpus is being cleaned.

```
corpusTweet <- tm_pre_process(contentTweet)
corpusNews <- tm_pre_process(contentNews)
corpusBlog <- tm_pre_process(contentBlog)
corpusPart <- tm_pre_process(contentPart)
```

2.4 Tokenization of data

Tokenization function generates dataset based on given token count. Multiple token count is refereed as per below

- 1 token : UniGram
- 2 token : BiGram
- 3 token : TriGram
- 4 token : QuadGram

Tokenization function also provides top entries where maximum frequency of tokens are present.

```
#tokenizer function
tokenizer <- function(corpusTitle, tokenCount) {
        token <- NGramTokenizer(
                corpusTitle,
                Weka_control(
                        min = tokenCount,
                        max = tokenCount,
                        delimiters = " \\r\\n\\t.,;:\"()?!"
                )
        )
        token
}

#function to generate top values from given token count
gramTopCount <- function(corpusTitle, tokenCount, TopCount = 0) {
        token <- tokenizer(corpusTitle, tokenCount)
        gram <- data.frame(table(token))
        gram <- gram[order(gram$Freq, decreasing = T), ]
        rownames(gram) <- NULL
        colnames(gram) <- c("Word", "Frequency")
        if(TopCount > 0){
                gram <- gram[1:TopCount, ]
                print(gramPlot(gram))
        }
        gram
}
```

Using above function, corpus of various data is tokenized for further visualization & summary statistics.

```
#Generate unigram, bigram, trigram & quadgram for blog dataset
blog.unigram <- gramTopCount(corpusBlog, 1)
blog.digram <- gramTopCount(corpusBlog, 2)
blog.trigram <- gramTopCount(corpusBlog, 3)
blog.quadgram <- gramTopCount(corpusBlog, 4)
#Generate unigram, bigram, trigram & quadgram for news dataset
news.unigram <- gramTopCount(corpusNews, 1)
news.digram <- gramTopCount(corpusNews, 2)
news.trigram <- gramTopCount(corpusNews, 3)
news.quadgram <- gramTopCount(corpusNews, 4)
#Generate unigram, bigram, trigram & quadgram for tweet dataset
tweet.unigram <- gramTopCount(corpusTweet, 1)
tweet.digram <- gramTopCount(corpusTweet, 2)
tweet.trigram <- gramTopCount(corpusTweet, 3)
tweet.quadgram <- gramTopCount(corpusTweet, 4)
#Generate unigram, bigram, trigram & quadgram for all dataset
part.unigram <- gramTopCount(corpusPart, 1)
part.digram <- gramTopCount(corpusPart, 2)
part.trigram <- gramTopCount(corpusPart, 3)
part.quadgram <- gramTopCount(corpusPart, 4)
```

3 Visualization

Visualization of various aspect data is key aspect of exploratory data analysis. In this chapter, various visualizations are used to provide insight in to data. Histogram, Unique word analysis for 50% & 90% coverage and WordCloud methodologies are used.

3.1 Histogram

Histograms shows representation of the distribution of data. In context o this analysis, histogram shows the frequency of occurrence of various words & its combinations. This analysis is useful to understand distribution of words and pattern of various n-grams.

3.1.1 Function

To use the repetitive analysis on various n-grams, simple function is designed to calculate histogram with top entries (configurable) from 4 corpus data (tweet, blog, news, combination). This function generates 4 histogram and then arranges them in panel of 2x2 for better visualization.

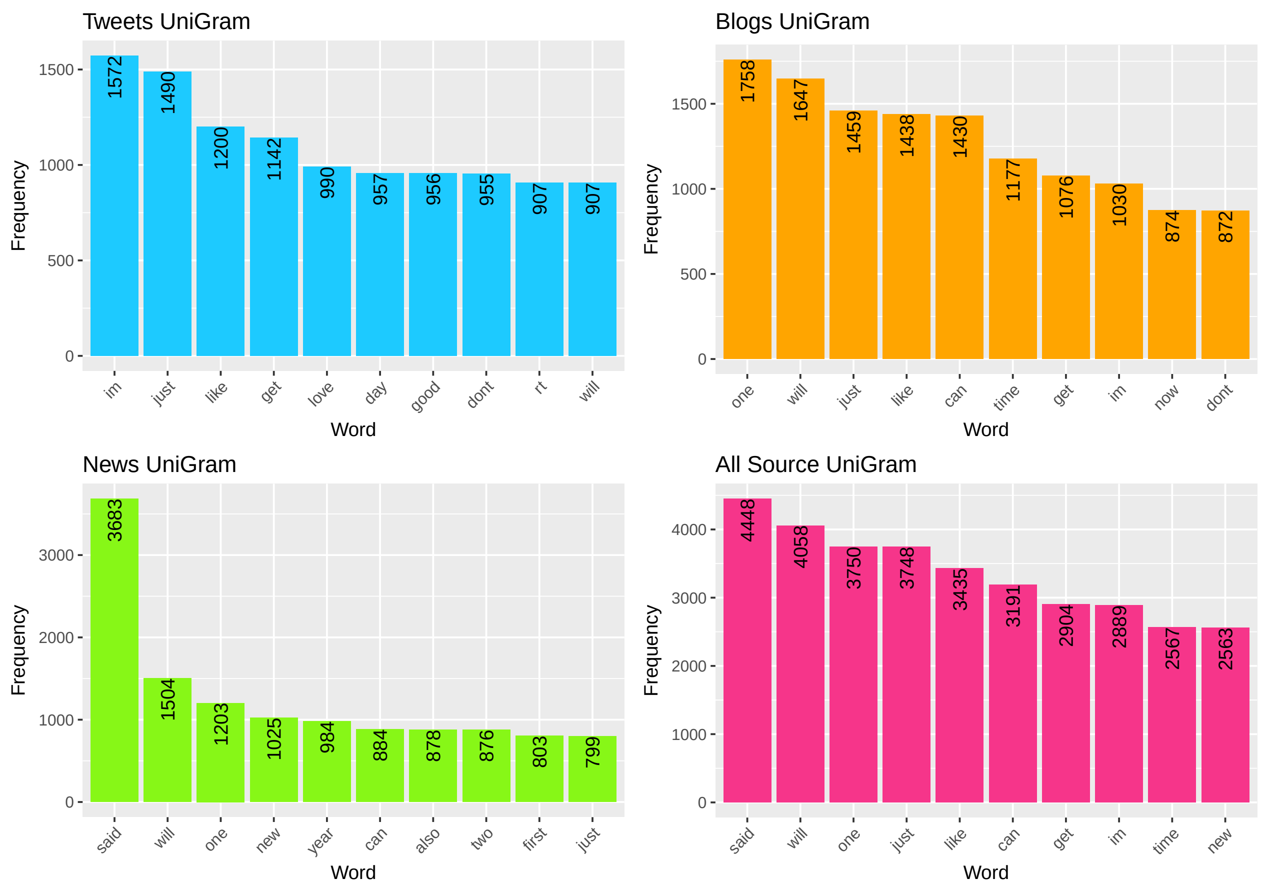
```
gramPlotQuad <-
        function(tweetgram,
                 bloggram,
                 newsgram,
                 partgram,
                 title = "NoGram",
                 TopCount = 10) {
                gtweet <-
                        ggplot(head(tweetgram, TopCount),
                               aes(x = reorder(Word, -Frequency), y = Frequency)) +
                        geom_bar(stat = "Identity", fill = "#1dcaff") +
                        geom_text(aes(label = Frequency),
                                  hjust = 1,
                                  angle = 90) +
                        xlab("Word") +
                        ggtitle(paste("Tweets", title)) +
                        theme(axis.text.x = element_text(angle = 45, hjust = 1))
                gblog <-
                        ggplot(head(bloggram, TopCount),
                               aes(x = reorder(Word, -Frequency), y = Frequency)) +
                        geom_bar(stat = "Identity", fill = "orange") +
                        geom_text(aes(label = Frequency),
                                  hjust = 1,
                                  angle = 90) +
                        xlab("Word") +
                        ggtitle(paste("Blogs", title)) +
                        theme(axis.text.x = element_text(angle = 45, hjust = 1))
                gnews <-
                        ggplot(head(newsgram, TopCount),
                               aes(x = reorder(Word, -Frequency), y = Frequency)) +
                        geom_bar(stat = "Identity", fill = "#87F717") +
                        geom_text(aes(label = Frequency),
                                  hjust = 1,
                                  angle = 90) +
                        xlab("Word") +
                        ggtitle(paste("News", title)) +
                        theme(axis.text.x = element_text(angle = 45, hjust = 1))
                gall <-
                        ggplot(head(partgram, TopCount),
                               aes(x = reorder(Word, -Frequency), y = Frequency)) +
                        geom_bar(stat = "Identity", fill = "#F6358A") +
                        geom_text(aes(label = Frequency),
                                  hjust = 1,
                                  angle = 90) +
                        xlab("Word") +
                        ggtitle(paste("All Source", title)) +
                        theme(axis.text.x = element_text(angle = 45, hjust = 1))
                grid.arrange(gtweet,
                             gblog,
                             gnews,
                             gall,
                             ncol = 2,
                             nrow = 2)

        }
```

3.1.2 Visualization

In this chapter, previously defined histogram function is used to generate analysis for Unigram, Bigram, Trigram & Quadgram for top 10 entries.

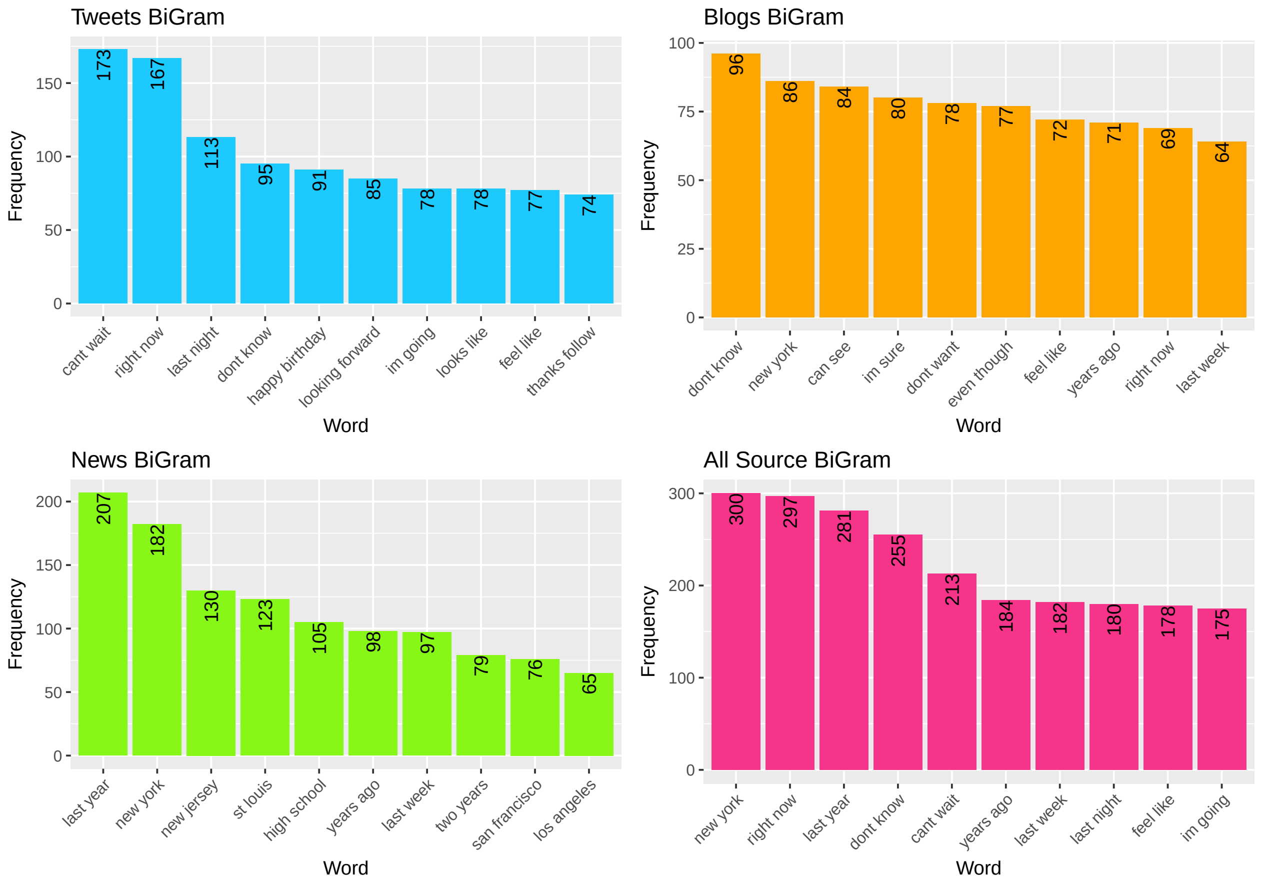
3.1.2.1 Top 10 UniGrams across all datasets

```
##ploting
g1 <- gramPlotQuad(
        tweetgram = tweet.unigram,
        bloggram = blog.unigram,
        newsgram = news.unigram,
        partgram = part.unigram,
        title = "UniGram"
)
```



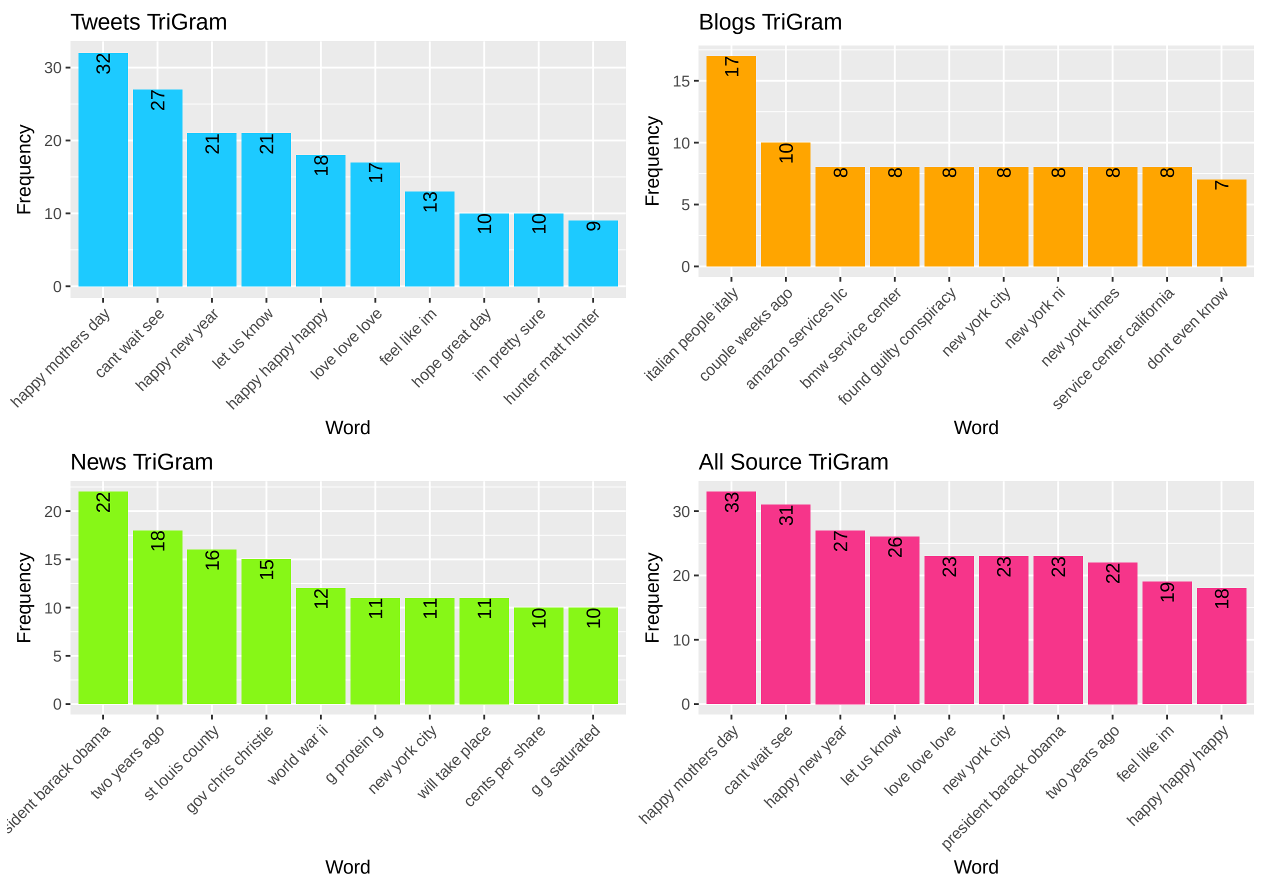
3.1.2.2 Top 10 Bigrams across all datasets

```
##ploting
g2 <- gramPlotQuad(
        tweetgram = tweet.digram,
        bloggram = blog.digram,
        newsgram = news.digram,
        partgram = part.digram,
        title = "BiGram"
)
```



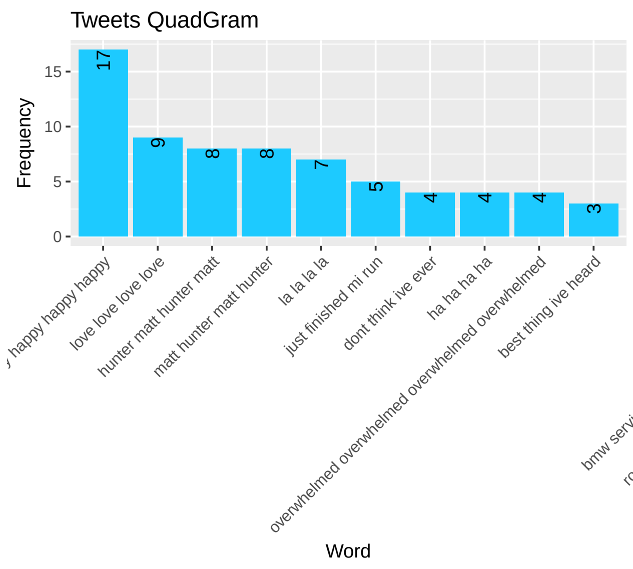
3.1.2.3 Top 10 TriGrams across all datasets

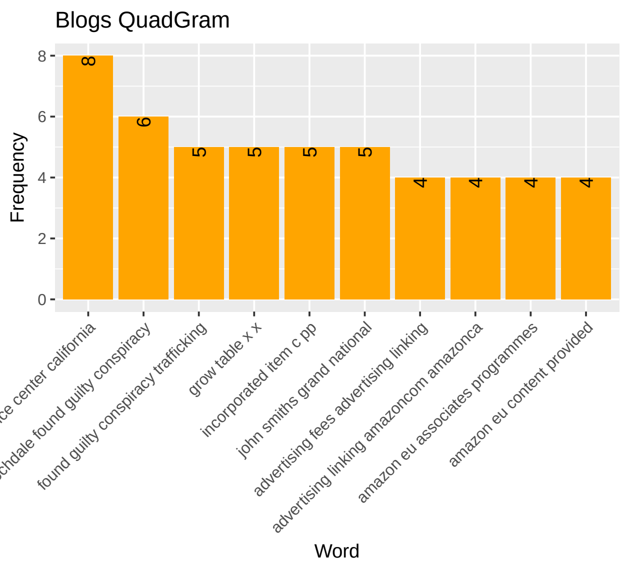
```
##ploting
g3 <- gramPlotQuad(
        tweetgram = tweet.trigram,
        bloggram = blog.trigram,
        newsgram = news.trigram,
        partgram = part.trigram,
        title = "TriGram"
)
```

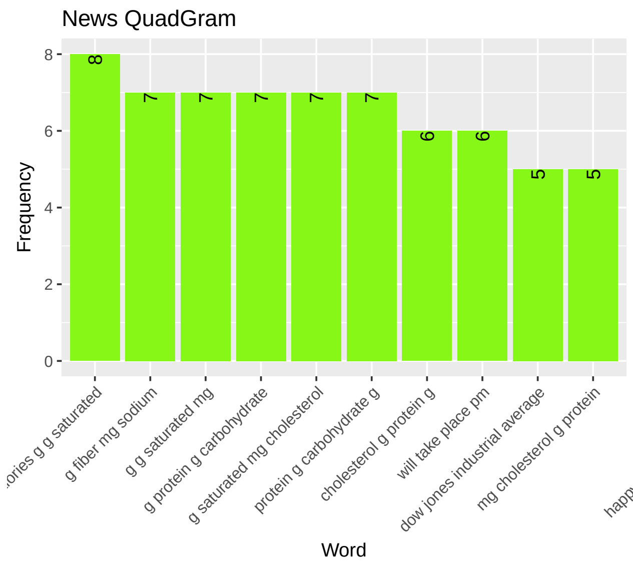


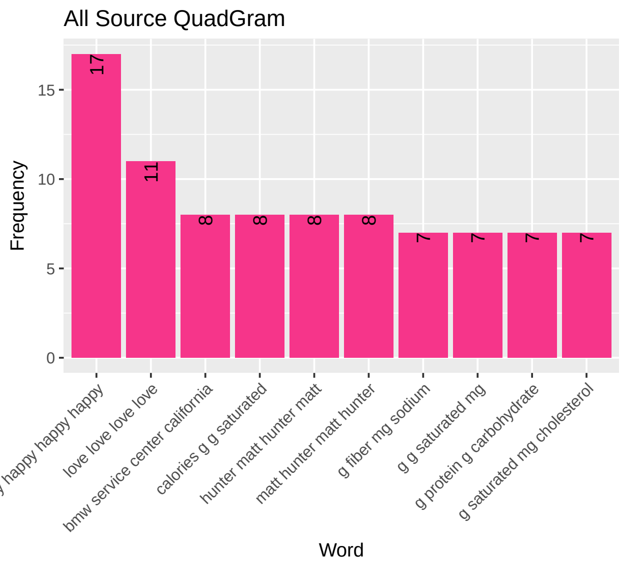
3.1.2.4 Top 10 QuadGrams across all datasets

```
##ploting
g4 <- gramPlotQuad(
        tweetgram = tweet.quadgram,
        bloggram = blog.quadgram,
        newsgram = news.quadgram,
        partgram = part.quadgram,
        title = "QuadGram"
)
```









3.1.3 Discussion

- For Unigram, news corpus distribution dominates combined corpus. Pattern of news , blogs & tweet corpus is different. Combined corpus shows aggregate pattern which is weighted average of 3 corpus.
- For Bigram, no corpus distribution dominates the combined corpus. Frequency of occurrence of bigrams are ~10 times lesser then unigrams which is normal in nature.
- For Trigram, tweet corpus distribution dominates combined corpus. Frequency of occurrence of bigrams are ~10 times lesser then bigrams and ~100 times lesser then unigrams.
- For Quadgrams, tweet corpus distribution dominates combined corpus. However, frequencies are so small and conclusion is not reliable.

3.2 Unique Word Analysis for 50% and 90% coverage

Unique word analysis is technique derived from Cumulative Distribution Function. Here each unique word/n-gram is assigned a probability based on occurrence in given corpus and then CDF of same is calculated. 50% coverage means, how many unique words are needed to represent 50% of corpus and similarly 90% coverage means how many unique words are needed to represent 90% of corpus. This analysis, provides an insight about how much words are needed to cover part of corpus.

3.2.1 Function

To use the repetitive analysis on various n-grams, simple function is designed to calculate coverage of 50% and 90% from 4 corpus data (tweet, blog, news, combination). This function generates 4 CDF and then arranges them in panel of 2x2 for better visualization. 50% coverage related lines are drawn in green and 90% related lines are drawn in blue. Exact number of entries needed are displayed in title of each figure for 50% & 90% coverage.

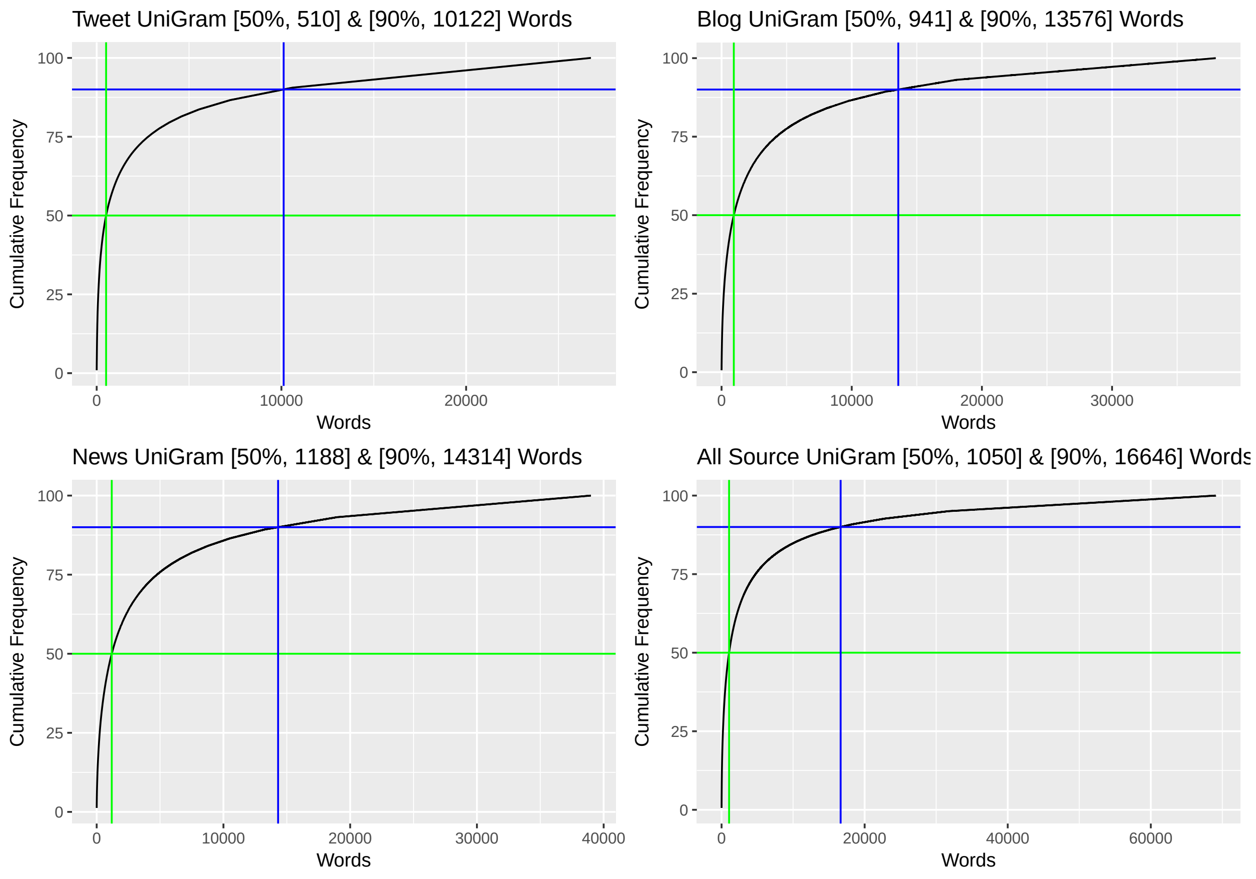
```r
plot5090interval <- function(gram, title = "") {
        total <- sum(gram$Frequency)
        gram$cumFreq <- 100 * cumsum(gram$Frequency) / total
        yintercept50 <- 50
        yintercept90 <- 90
        xintercept50 = sum(gram$cumFreq < yintercept50)
        xintercept90 = sum(gram$cumFreq < yintercept90)
        g <-
                ggplot(gram, aes(y = cumFreq, x = seq_along(cumFreq))) +
                geom_line() +
                geom_hline(yintercept = yintercept50, col = "green") +
                geom_hline(yintercept = yintercept90, col = "blue") +
                geom_vline(xintercept = xintercept50, col = "green") +
                geom_vline(xintercept = xintercept90, col = "blue") +
                xlab("Words") +
                ylab("Cumulative Frequency") +
                ggtitle(paste0(
                        title,
                        " [50%, ",
                        xintercept50,
                        "] & [90%, ",
                        xintercept90,
                        "] Words"
                ))
        g
}

Plot5090intervalQuad <- function(tweetgram,
                                 bloggram,
                                 newsgram,
                                 partgram,
                                 title = "NoGram") {
        g1 <- plot5090interval(tweetgram, title = paste0("Tweet ", title))
        g2 <-
                plot5090interval(bloggram, title = paste0("Blog ", title))
        g3 <-
                plot5090interval(newsgram, title = paste0("News ", title))
        g4 <-
                plot5090interval(partgram, title = paste0("All Source ", title))
        grid.arrange(g1, g2, g3, g4, ncol = 2, nrow = 2)
}
```

3.2.2 Visualization

In this chapter, previously defined coverage function is used to generate analysis for Unigram, Bigram, Trigram & Quadgram with 50% & 90% coverage.

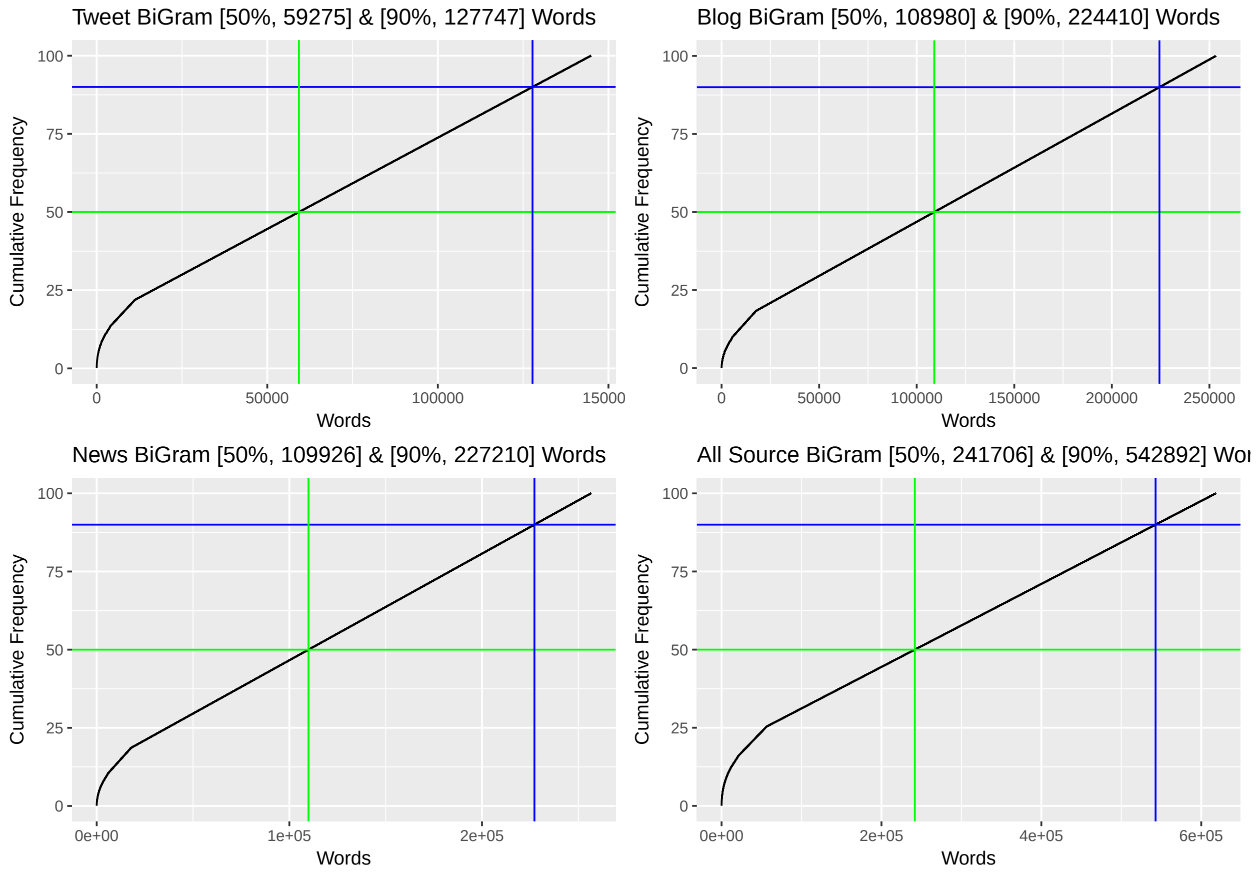
3.2.2.1 Coverage of UniGrams across all datasets

```
##ploting
p1 <- Plot5090intervalQuad(
        tweetgram = tweet.unigram,
        bloggram = blog.unigram,
        newsgram = news.unigram,
        partgram = part.unigram,
        title = "UniGram"
)
```



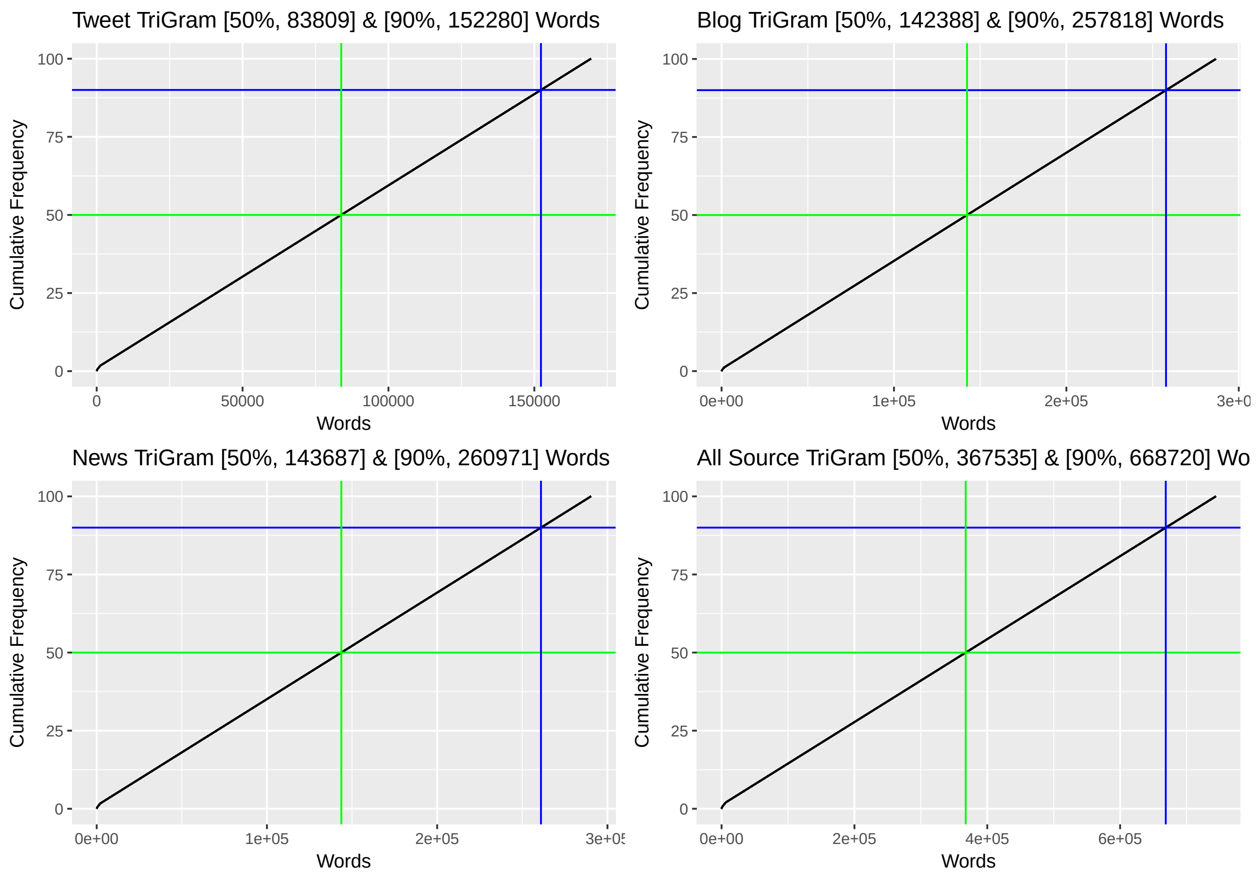
3.2.2.2 Coverage of Bigrams across all datasets

```
##ploting
p2 <- Plot5090intervalQuad(
        tweetgram = tweet.digram,
        bloggram = blog.digram,
        newsgram = news.digram,
        partgram = part.digram,
        title = "BiGram"
)
```



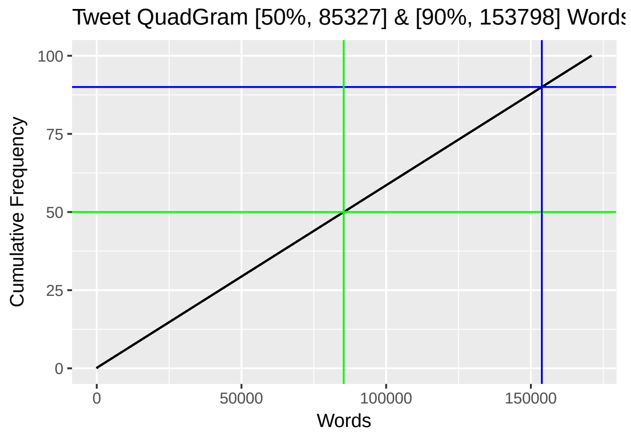
3.2.2.3 Coverage of TriGrams across all datasets

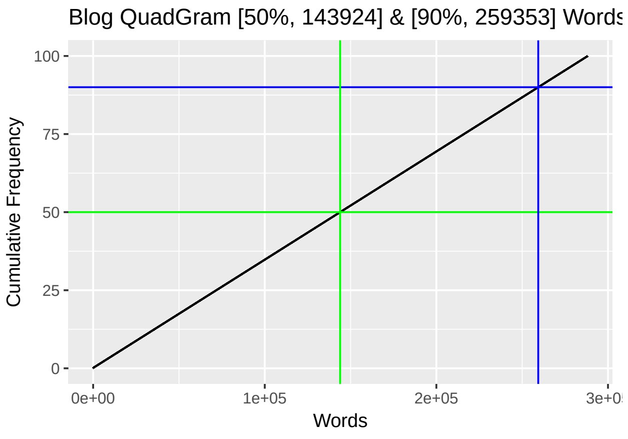
```
##ploting
p3 <- Plot5090intervalQuad(
        tweetgram = tweet.trigram,
        bloggram = blog.trigram,
        newsgram = news.trigram,
        partgram = part.trigram,
        title = "TriGram"
)
```

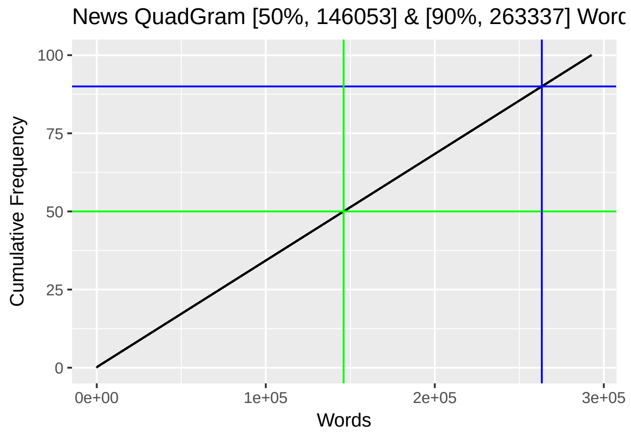


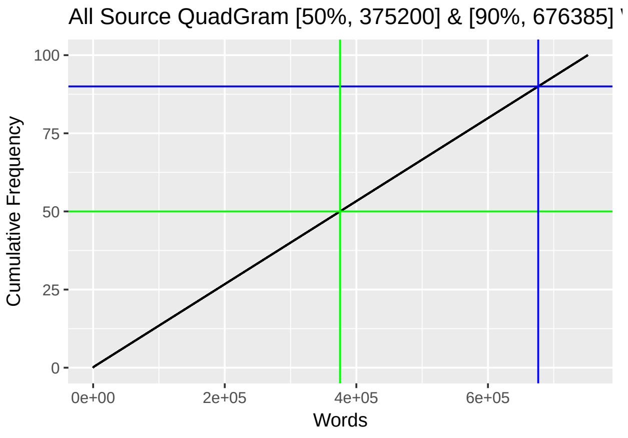
3.2.2.4 Coverage of QuadGrams across all datasets

```
##ploting
p4 <- Plot5090intervalQuad(
        tweetgram = tweet.quadgram,
        bloggram = blog.quadgram,
        newsgram = news.quadgram,
        partgram = part.quadgram,
        title = "QuadGram"
)
```









3.2.3 Discussion

- For Unigram, all data corpus follows CDF curve which is representative of normal distribution function. However , tweet corpus needs only 510 words to represent 50% which is 2 times less of any other corpus. Similarly 90% coverage in tweet corpus is achieved by 10122 words which is 1.5 times less of any other corpus. Combined corpus shows aggregation effect.
- For Bigram, all data corpus follows unique CDF which is representative of normal distribution up to a limit and then represents uniform distribution. This means there are set of bigrams which are high on probability and remaining have uniform probability. Tweet corpus again stands out as 50% and 90% overage count is almost 8 times & 2.5 times less. Due to availability of uniform distribution, combined corpus shows surprising behavior. Combined corpus 50% & 90 % coverage comes at almost 1.75 times and 2.5 times higher than that of all individual corpus.
- For Trigram & Quadgram, all data corpus shows CDF pattern which is representative of uniform distribution only. Due to nature of uniform distribution, combined corpus shows behavior which is similar to that of bigrams.

3.3 WordCloud

A WordCloud is a visual representation of text data, typically used to depict keyword to visualize free form text. Usually single words (ngrams are possible) are used and importance of same is depicted by font size and color. This analysis provides an insight of prominent words/ngrams from content point of view and utilizes human visualization bandwidth to unearth the prominent patterns.

3.3.1 Function

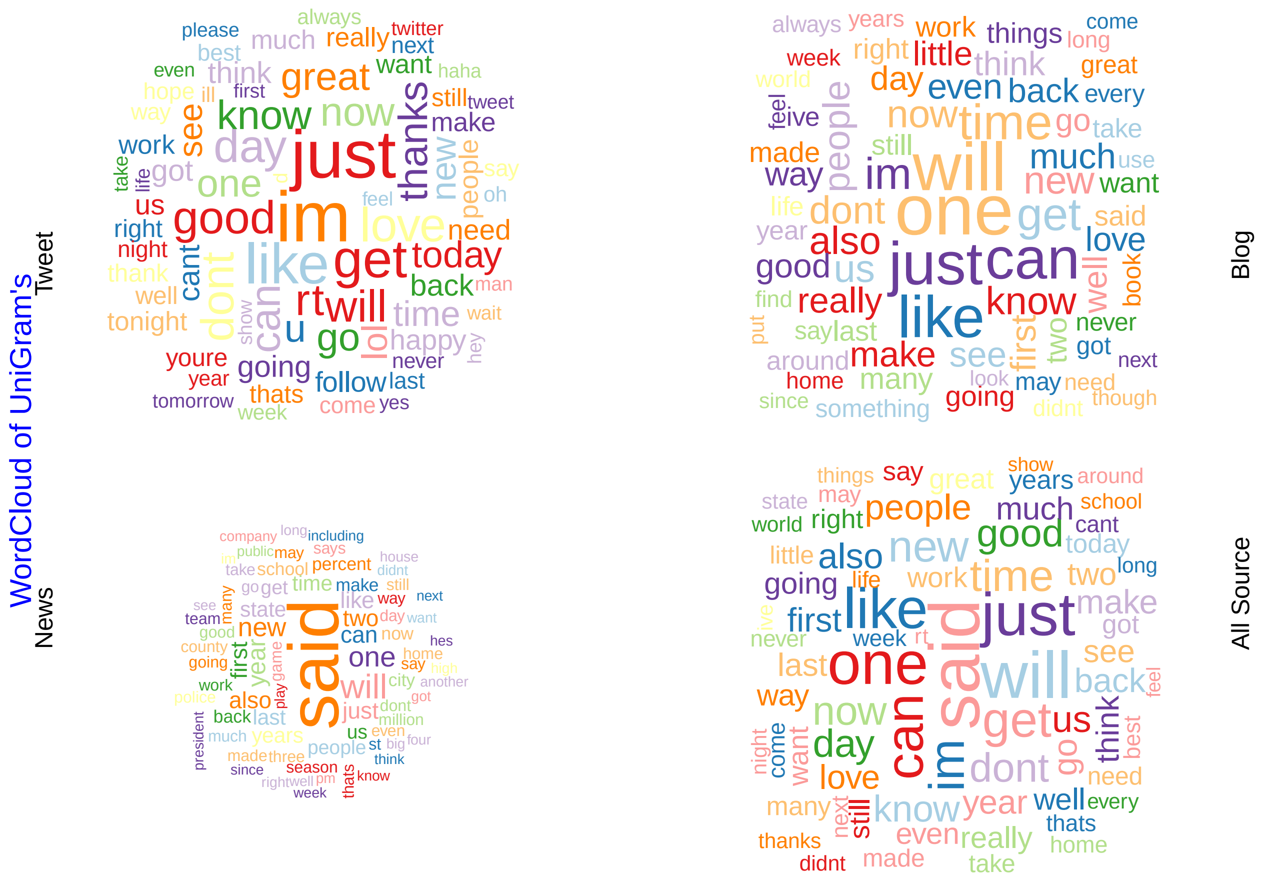
To use the repetitive analysis on single word & bigrams, simple function is designed to generate wordcloud from 4 corpus data (tweet, blog, news, combination). This function generates 4 wordcloud and then arranges them in panel of 2x2 for better visualization. At max, wordcloud can take show 75 entries provided each entry is occurring for more than 10 times in in corpus.

```r
cloudPlotQuad <-
        function(tweetgram,
                 bloggram,
                 newsgram,
                 partgram,
                 Type = "UniGram") {
                par(mfrow = c(2, 2))
                wordcloud(
                        as.character(tweetgram$Word),
                        tweetgram$Frequency,
                        min.freq = 10,
                        max.words = 75,
                        colors = brewer.pal(11, "Paired"),
                        random.order = F,
                        random.color = T,
                        rot.per = .15,
                        scale = c(4, 0.5)
                )
                wordcloud(
                        as.character(bloggram$Word),
                        bloggram$Frequency,
                        min.freq = 10,
                        max.words = 75,
                        colors = brewer.pal(11, "Paired"),
                        random.order = F,
                        random.color = T,
                        rot.per = .15,
                        scale = c(4, 0.5)
                )
                wordcloud(
                        as.character(newsgram$Word),
                        newsgram$Frequency,
                        min.freq = 10,
                        max.words = 75,
                        colors = brewer.pal(11, "Paired"),
                        random.order = F,
                        random.color = T,
                        rot.per = .15,
                        scale = c(4, 0.5)
                )
                wordcloud(
                        as.character(partgram$Word),
                        partgram$Frequency,
                        min.freq = 10,
                        max.words = 75,
                        colors = brewer.pal(11, "Paired"),
                        random.order = F,
                        random.color = T,
                        rot.per = .15,
                        scale = c(4, 0.5)
                )
                mtext(
                        paste0("WordCloud of ", Type, "'s"),
                        side = 2,
                        line = -2,
                        outer = TRUE,
                        col = "blue",
                        cex = 1.4
                )
                mtext(
                        "News                                          Tweet",
                        side = 2,
                        line = -3,
                        outer = TRUE,
                        col = "black",
                        cex = 1.2
                )
                mtext(
                        "All Source                                     Blog",
                        side = 4,
                        line = -2,
                        outer = TRUE,
                        col = "black",
                        cex = 1.2
                )

                par(mfrow = c(1, 1))
        }
```

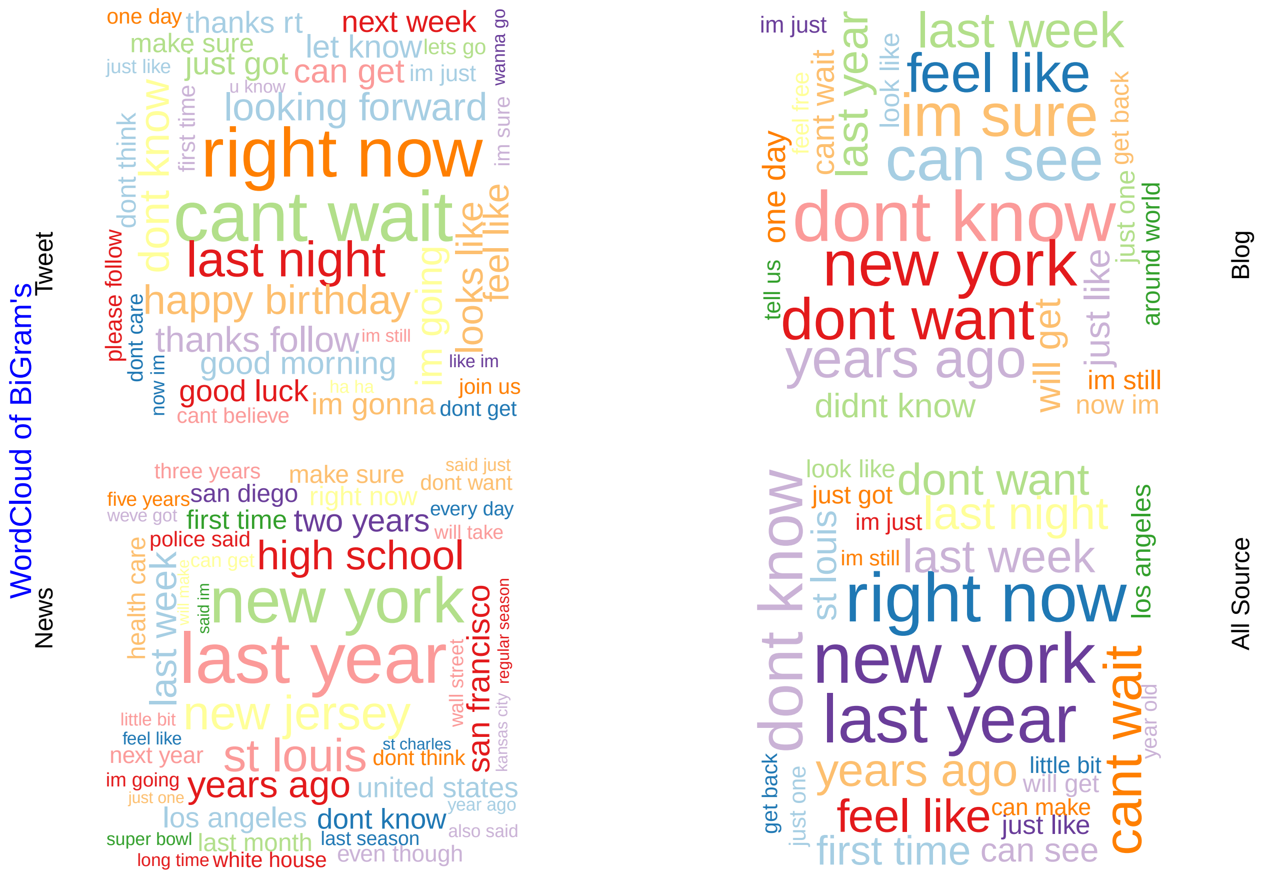
3.3.2 Visualization

In this chapter, previously defined wordcloud function is used to generate wordcloud for Unigram, Bigram.

3.3.2.1 WordCloud of UniGrams across all datasets



3.3.2.2 WordCloud of BiGrams across all datasets



3.3.3 Discussion

- For Unigram, wordcloud of all corpus provides a stunning differences in terms of occurrences. Also note that top 10 entries from histogram functions are in center. News wordcloud reveals that one specific word “said” is dominating in nature. This is not surprising as news article often cites other people’s opinion (like he/she said that . . .).
- For Bigram, wordcloud of all corpus is just expanded view of histogram. Intersting part is, due to normal distribution of some bigrams as identified in unique word coverage, combined corpus wordcloud is dominated by main bigrams from individual corpus. It reminds the superposition theorem.

4 Conclusion

- With a vocabulary of 1% of the total number of words we can predict 91% of the text.
- Twitter corpus has to be treated differently as characteristics of this corpus is different and in some cases heavily influences combined corpus. Hence in app, it is needed to provide twitter and non twitter mode
- In unigram & bigram, there exist a lot of low frequency terms which needs to be dropped. However trigram & quadgram have uniform distribution which means more terms with low frequency will be needed
- Train two different prediction algorithm based on n-grams with backtracking. One for twitter corpus only and one for other combined corpus
- Do not remove any words (including stop words or even profanity words) from the list in order to avoid creating “holes” in the data that could lead to incorrect n-grams and incorrect prediction -Ensure that

profanity words are not predicted at all (existing approach to remove profanity words from corpus is sufficient)

- Data corpus is huge and we intend to design lightweight application which means during modeling care needs to be taken on run time & memory consumption. Additionally trade off with accuracy need to be done carefully to improve overall usability of app

5 Reproducibility

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] wordcloud_2.5      RColorBrewer_1.1-2 gridExtra_2.2.1
## [4] ggplot2_2.2.1      RWeka_0.4-30       SnowballC_0.5.1
## [7] tm_0.6-2           NLP_0.1-9
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.9       knitr_1.15.1      magrittr_1.5
##  [4] RWekajars_3.9.1-1 munsell_0.4.3     colorspace_1.3-2
##  [7] highr_0.6         stringr_1.1.0     plyr_1.8.4
## [10] tools_3.3.2       parallel_3.3.2    grid_3.3.2
## [13] gtable_0.2.0      htmltools_0.3.5   assertthat_0.1
## [16] lazyeval_0.2.0    yaml_2.1.14       rprojroot_1.1
## [19] digest_0.6.11     tibble_1.2        rJava_0.9-8
## [22] evaluate_0.10     slam_0.1-40       rmarkdown_1.3
## [25] stringi_1.1.2     scales_0.4.1      backports_1.0.4
```

Source & relevant files can be found at git hub repository : https://github.com/parmarmanojkumar/DataScience_CapStone.git