Scratchpad

Manojkumar Parmar

1/23/2017

Contents

1 Environment **1**

2 Procesing Data **2**
2.1 Fetching . . . 2
2.2 Cleaning Data . . . 2
2.3 Tokenization of data . . . 4

3 Visualization **5**
3.1 Histogram . . . 5
3.1.1 Function . . . 5
3.1.2 Visualization . . . 6
3.1.2.1 Top 10 UniGrams across all datasets . . . 6
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 . . . 10
3.2.2 Visualization . . . 11
3.2.2.1 Coverage of UniGrams across all datasets . . . 11
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 . . . 15
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 . . . 17
3.3.3 Discussion . . . 18

4 Conclusion **18**

1 Environment

```
#load libraries
library(datasets)
library(tm)
library(SnowballC)
library(RWeka)
library(ggplot2)
library(gridExtra)
library(wordcloud)
```

```
#cleanup environmant
rm(list = ls())
cat("\014")
```

2 Procesing Data

2.1 Fetching

```
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")

}
```

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)
```

2.2 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, lets clean all corpus.

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

2.3 Tokenization of data

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

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

Tokenization function also provides top enties where maximum freuqncy 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, lets tokenize various corpses.

```
#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

3.1 Histogram

3.1.1 Function

```
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),
```

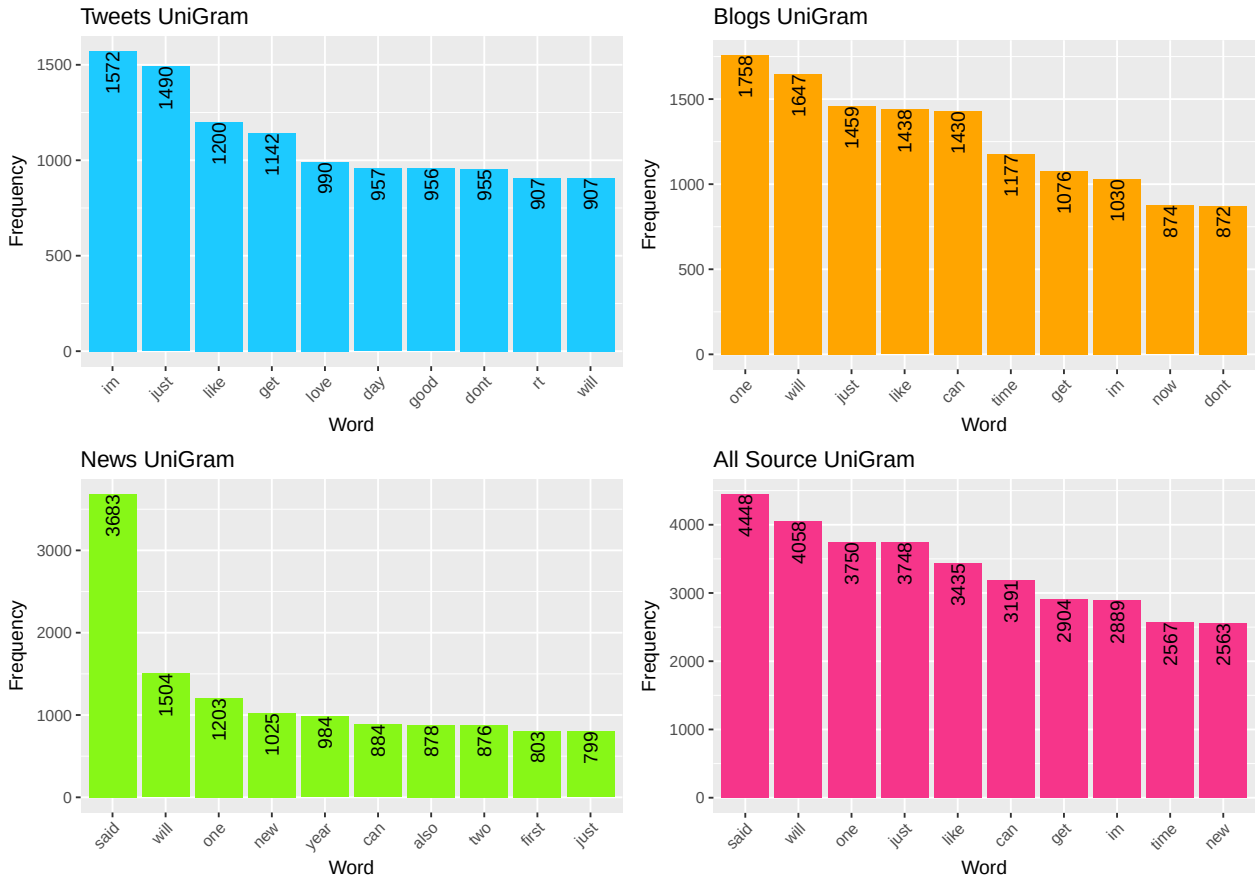
```
                                aes(x = reorder(Word, -Frequency), y = Frequency)) +
                        geom_bar(stat = "Identity", fill = "#F6358A") +
                        geom_text(aes(label = Frequency),
                                  hjust = 1,
                                  angle = 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

Lets Plot the graphs

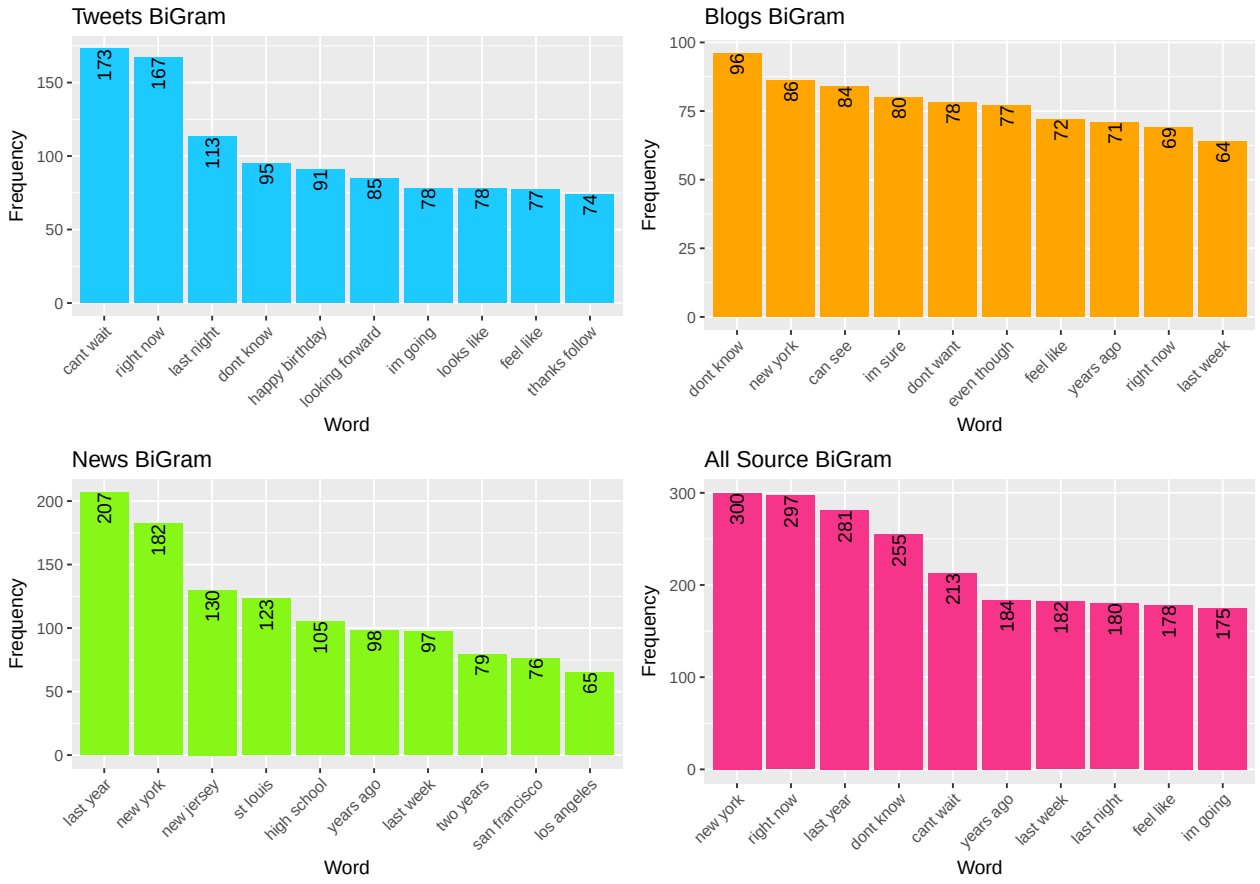
3.1.2.1 Top 10 UniGrams across all datasets

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



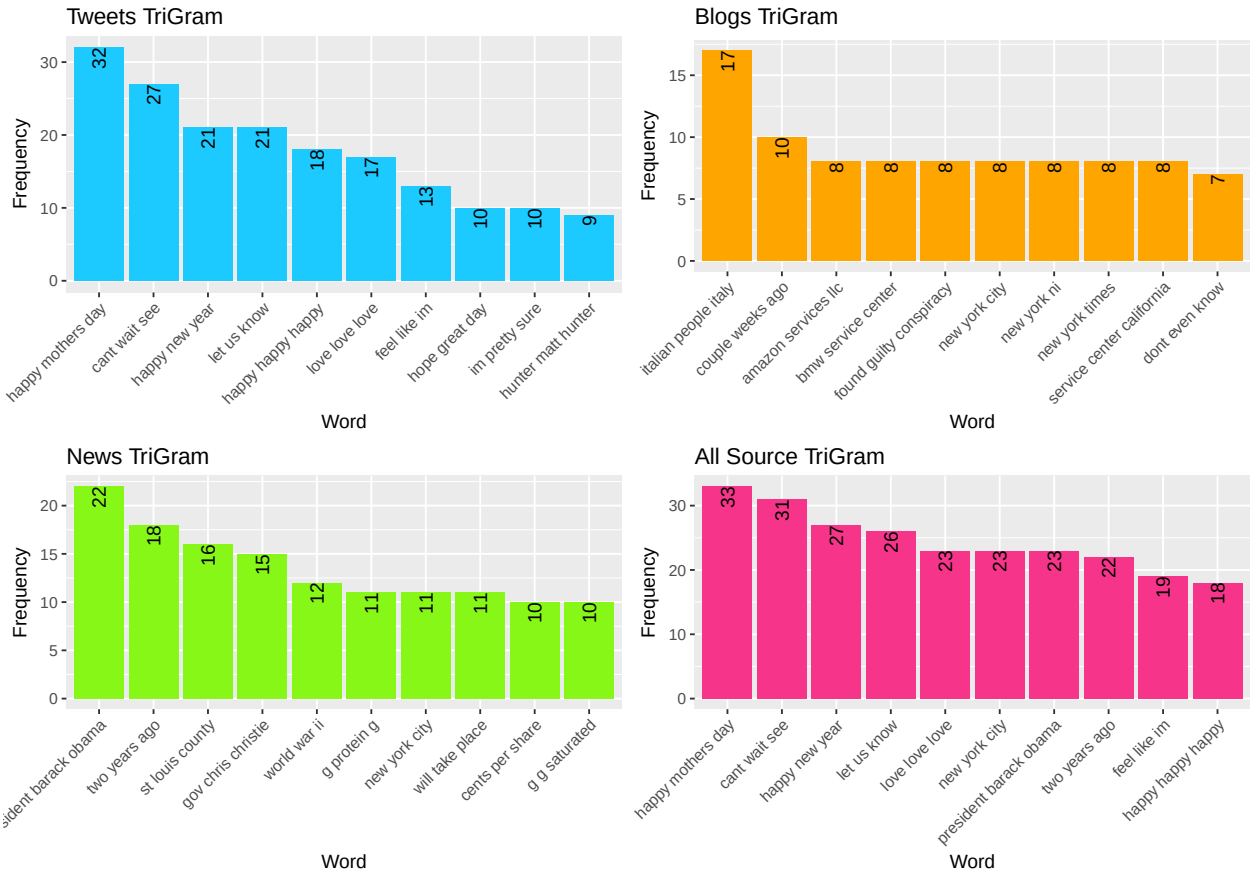
3.1.2.2 Top 10 Bigrams across all datasets

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



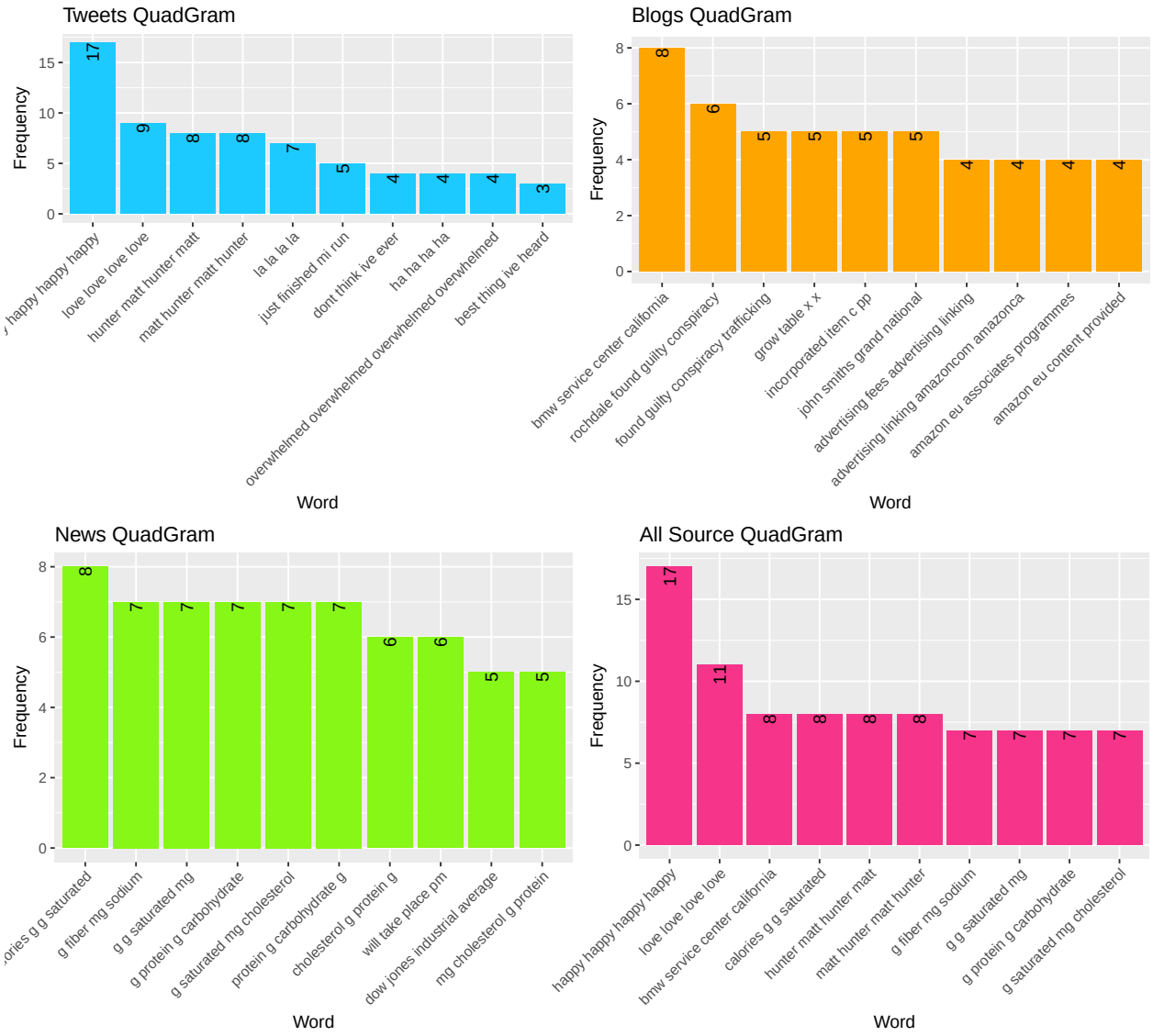
3.1.2.3 Top 10 TriGrams across all datasets

```
##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

All application needs different type of modeling as distribution of various grams differes.

3.2 Unique Word Analysis for 50% and 90% coverage

3.2.1 Function

```
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 <-
```

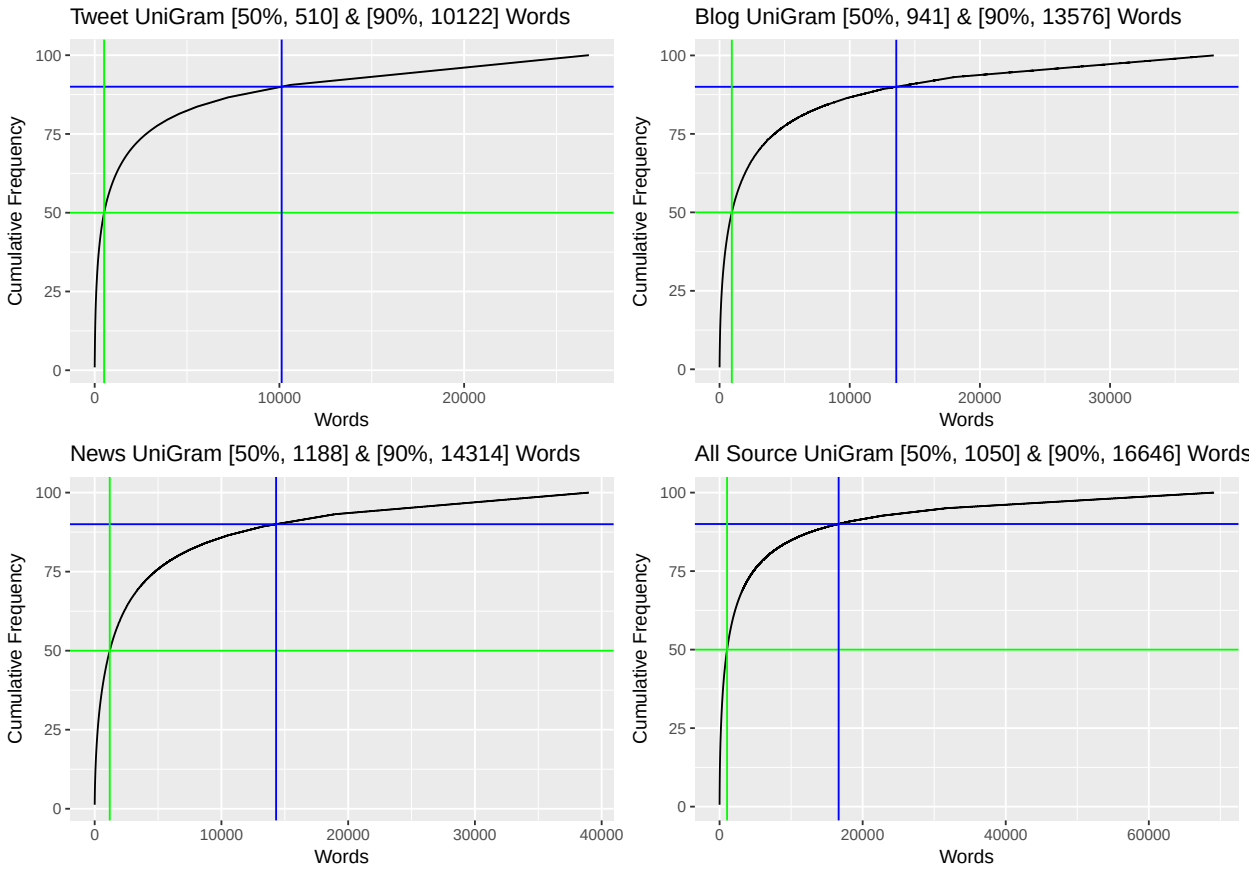
```
                ggplot(gram, aes(y = cumFreq, x = seq_along(cumFreq))) +
                geom_line() +
                geom_hline(yintercept = yintercept50, col = "green") +
                geom_hline(yintercept = yintercept90, col = "blue") +
                geom_vline(xintercept = xintercept50, 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

Lets Plot the graphs

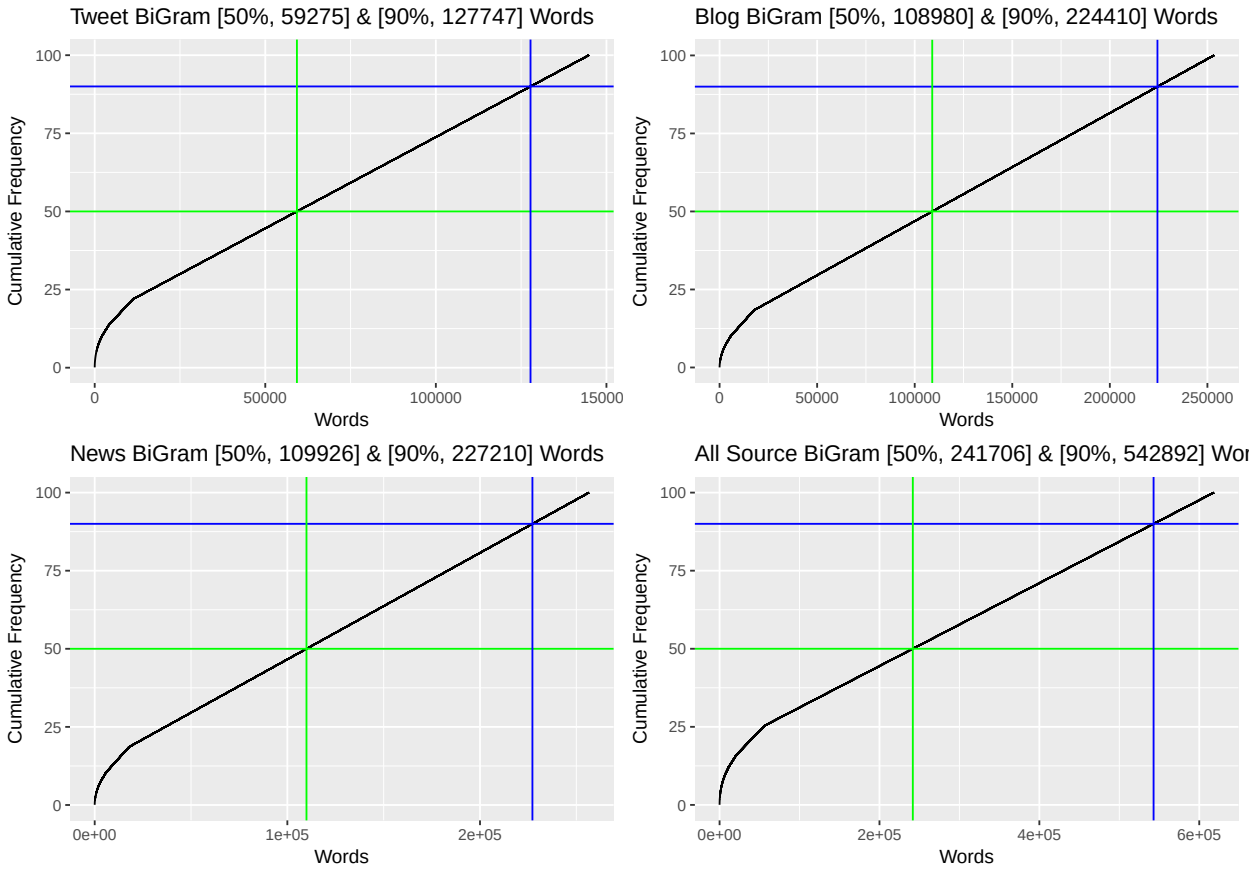
3.2.2.1 Coverage of UniGrams across all datasets

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



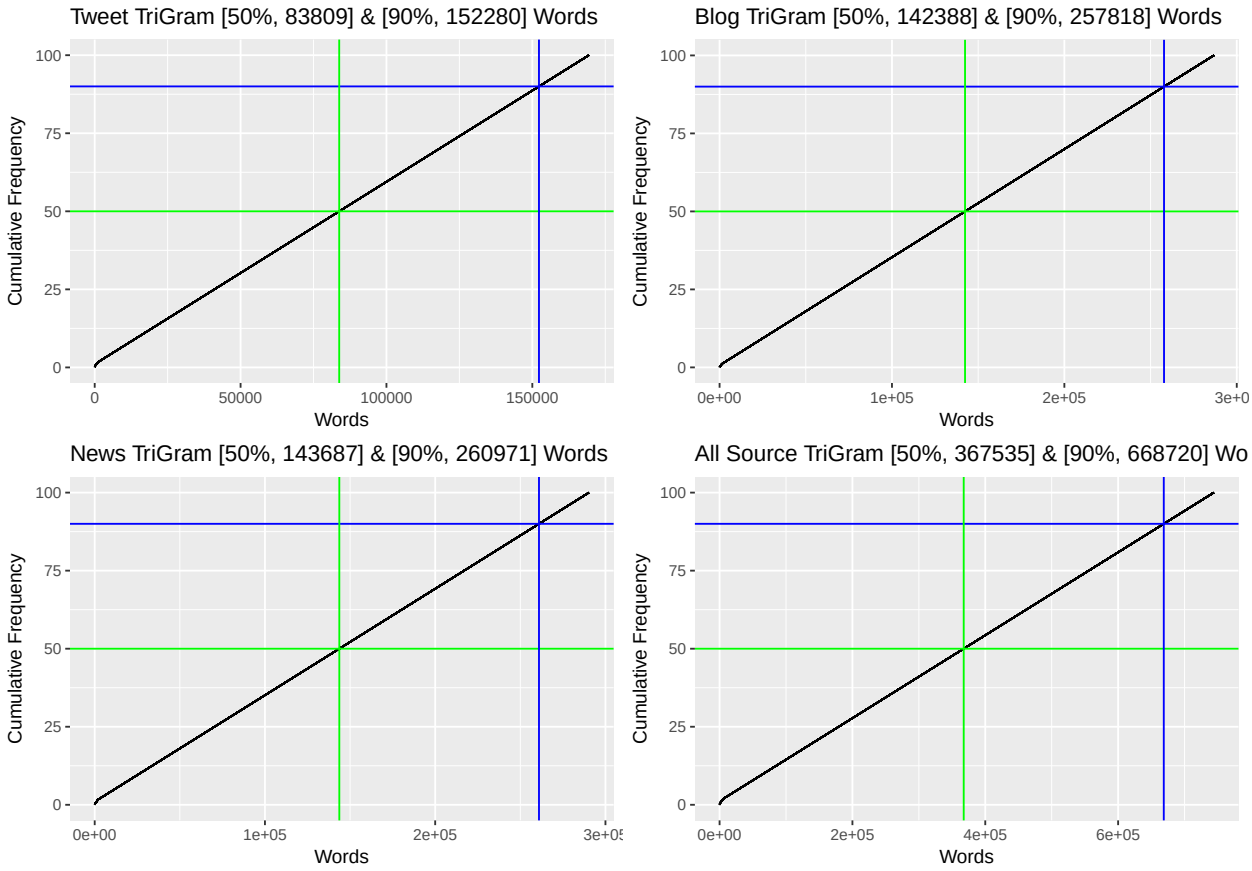
3.2.2.2 Coverage of Bigrams across all datasets

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



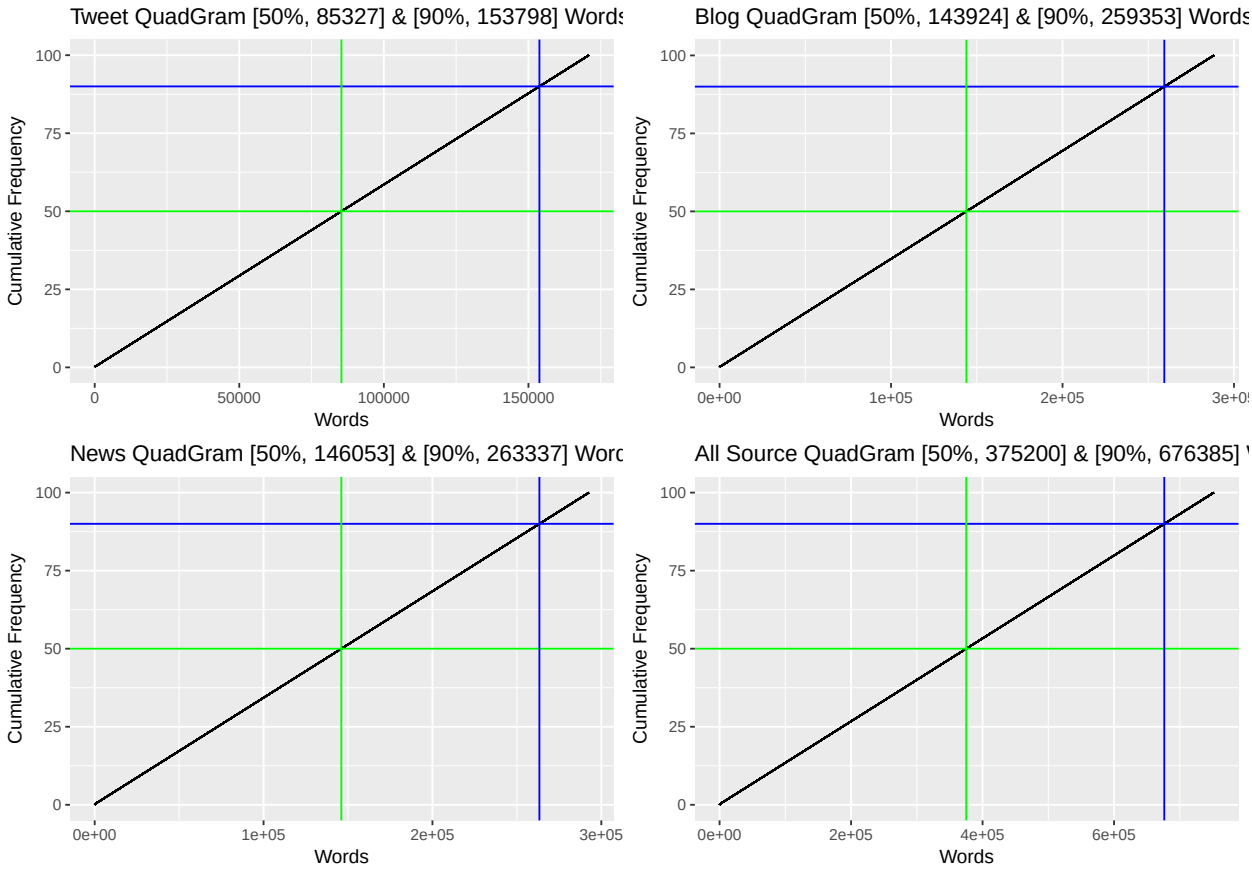
3.2.2.3 Coverage of TriGrams across all datasets

```
##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

All application needs different type of modeling as distribution of various grams differes.

3.3 WordCloud

3.3.1 Function

```
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,
```

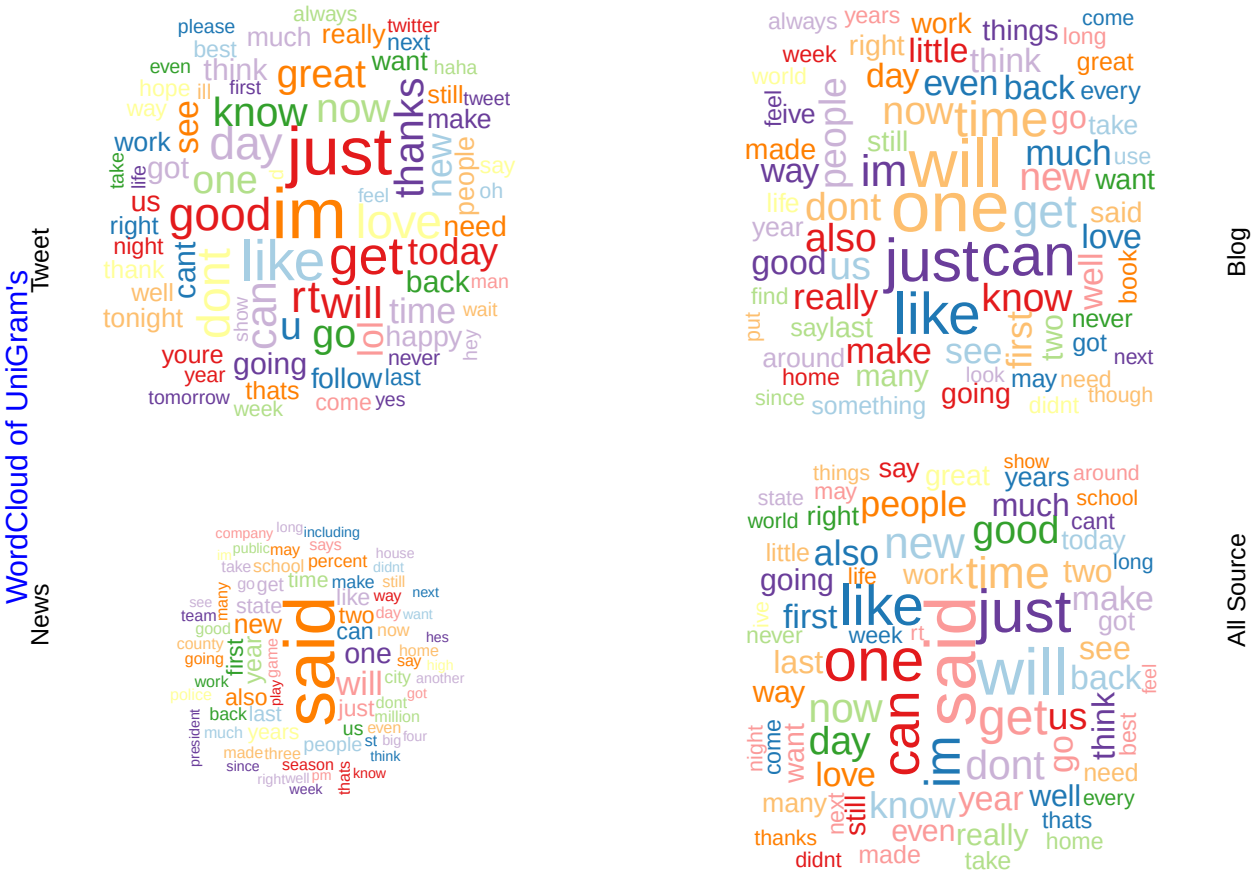
```
        scale = c(4, 0.5)
    )
    wordcloud(
        as.character(bloggram$Word),
        bloggram$Frequency,
        min.freq = 10,
        max.words = 75,
        colors = brewer.pal(11, "Paired"),
        random.order = F,
        random.color = T,
        rot.per = .15,
        scale = c(4, 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))
        }
```

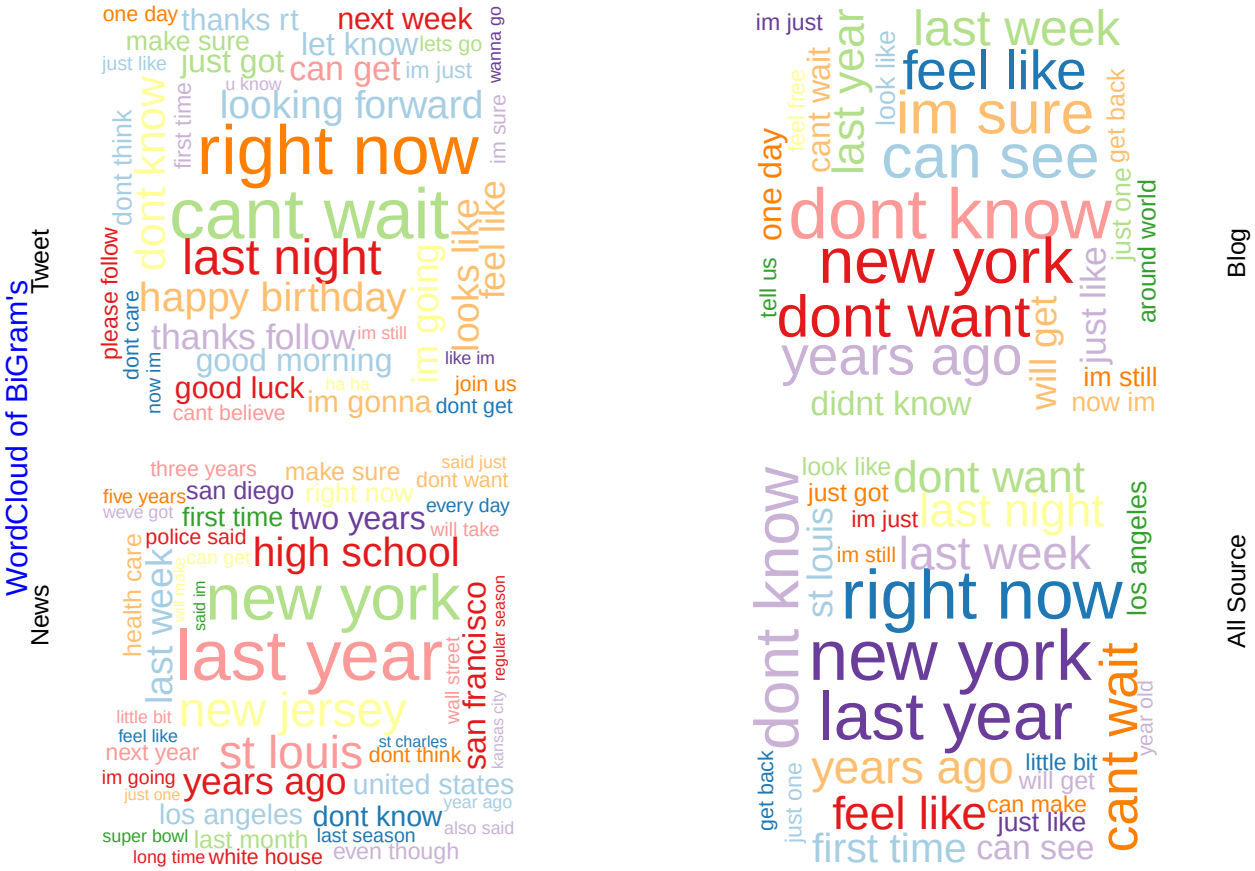
3.3.2 Visualization

Lets Plot th graphs

3.3.2.1 WordCloud of UniGrams across all datasets



3.3.2.2 WordCloud of BiGrams across all datasets



3.3.3 Discussion

All application needs different type of modeling as distribution of various grams differes.

4 Conclusion

Future application Devlopment.