Week 2 - Coursera Data Science Capstone

Michelle

April 3, 2018

Overview

The ultimate goal of this course is to build a predictive text model using natural language processing. The course is broken up by week, where this week, our milestone is to explore data in the news, blogs and twitter files. The data was read in, with basic things like the number of lines of text in each document and the number of words in each. To do harder analysis on the data, I needed to take much smaller samples of the data due to memory contraints (this could be addressed with higher memory computers if it were pursued at a higher level). With these smaller samples, I needed to clean and prepare the data to allow fore accurate analysis. The things I changed was I removed punctuation, made everything lower case (to avoid words like The and the being counted as two different words for example), and then I removed any non-ascii based characters.

After doing this, I was able to also look at the most frequent words found in these samples. Included below are the analysis of news, blogs and twitter respectively. Each of these sections have a plot of the top 10 most common words found in the sample, as well as an examination of something called "ngrams." N-grams are a sequence of words that are commonly seen in a sample of language data. I then look at the most common bigrams are (2 words in succession), and trigrams (3 words in succession). Plots of all of this information is found in the below data

Analysis

The datasets used are news articles, blog articles and twitter posts in the United States. Some of the first analysis should be figuring out the size and characteristics of the data.

1.Number of lines in each file

```
#1 How many lines in each file?
  con = file(paste0(docFiles,"en_US.news.txt"))
  news = readLines(con,warn=FALSE,encoding="UTF-8")
  close(con)
  newsLines = length(news)
  rm(con)
  #How many lines in US news?
  print(paste0("The number of lines in the US News document is: ",newsLines))
```

```
## [1] "The number of lines in the US News document is: 77259"
```

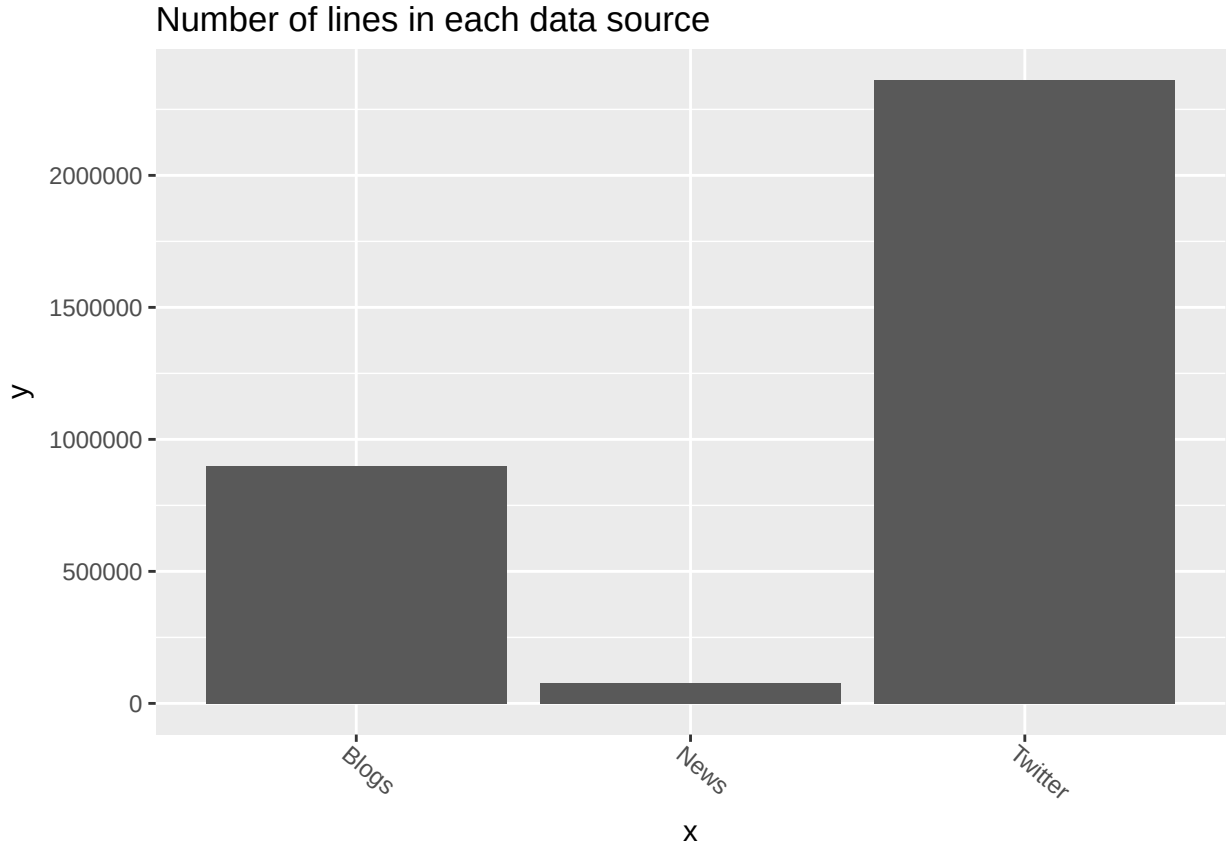
```
  con = file(paste0(docFiles,"en_US.blogs.txt"))
  blogs= readLines(con,warn=FALSE,encoding="UTF-8")
  close(con)
  rm(con)
  #how many lines in US blogs?
  blogLines = length(blogs)
  print(paste0("The number of lines in the US Blogs document is: ",blogLines))
```

```
## [1] "The number of lines in the US Blogs document is: 899288"
```

```
  con = file(paste0(docFiles,"en_US.twitter.txt"))
  tweets = readLines(con,warn=FALSE,encoding = "UTF-8")
  close(con)
  rm(con)
  #how many lines in US blogs?
  tweetLines = length(tweets)
  print(paste0("The number of lines in the US Blogs document is: ",tweetLines))
```

```
## [1] "The number of lines in the US Blogs document is: 2360148"
```

```
    line.df = data.frame(x=c("News","Blogs","Twitter"),y=c(newsLines,blogLines,tweetLines))
  ggplot(line.df,aes(x=x,y=y)) + ggtitle("Number of lines in each data source") + geom_bar(stat="identi
```




```
  gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  5031142 268.7    8273852 441.9  6861544 366.5
## Vcells 64014996 488.4   87226599 665.5 71034196 542.0
```

2. How many words in each file?

```
newsWordCount = wordcount(news)
 print(paste0("The number of words in the US News document is: ",newsWordCount))
```

```
## [1] "The number of words in the US News document is: 2643969"
```

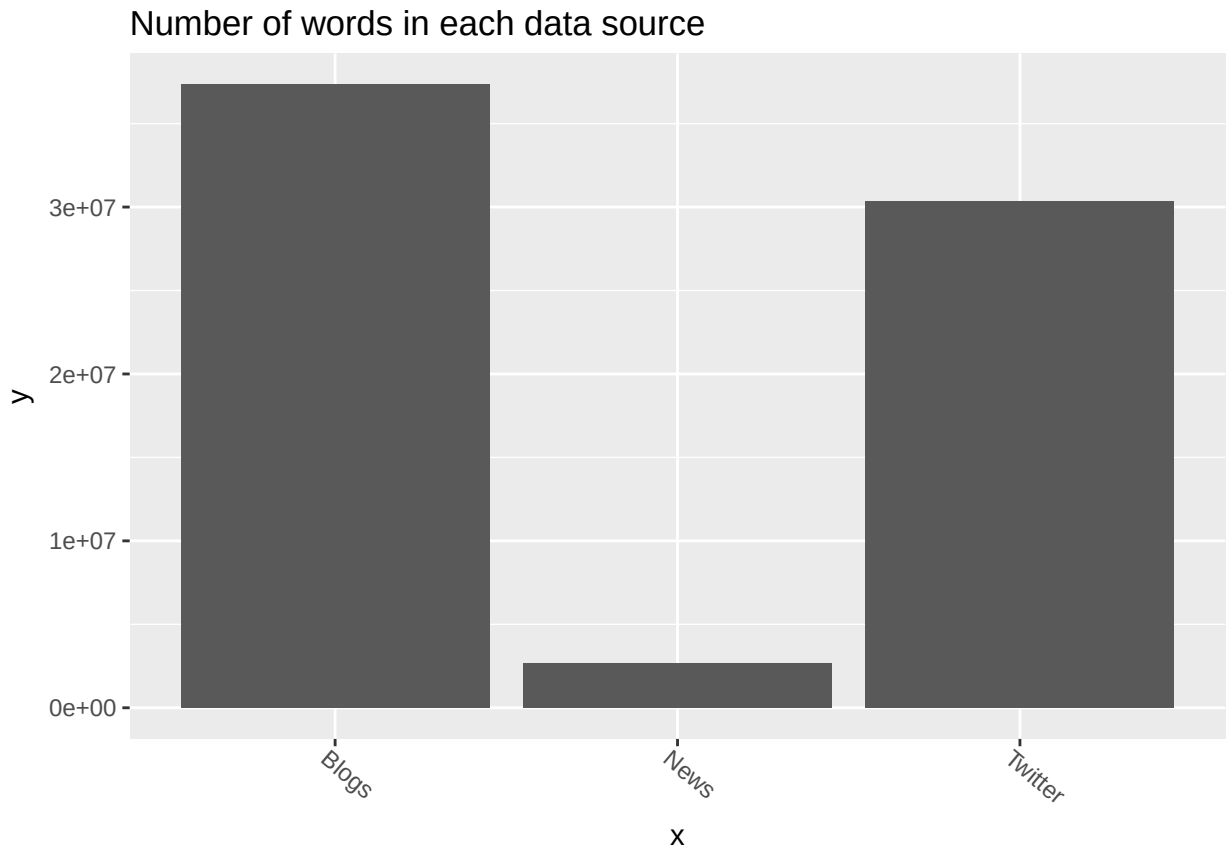
```
  blogsWordCount = wordcount(blogs)
  print(paste0("The number of words in the US Blogs document is: ",blogsWordCount))
```

```
## [1] "The number of words in the US Blogs document is: 37334131"
```

```
  twitterWordCount = wordcount(tweets)
  print(paste0("The number of words in the US Twitter document is: ",twitterWordCount))
```

```
## [1] "The number of words in the US Twitter document is: 30373543"
```

```
  word.df = data.frame(x=c("News","Blogs","Twitter"),y=c(newsWordCount,blogsWordCount,twitterWordCount))
  ggplot(word.df,aes(x=x,y=y)) + ggtitle("Number of words in each data source") + geom_bar(stat="identi
```

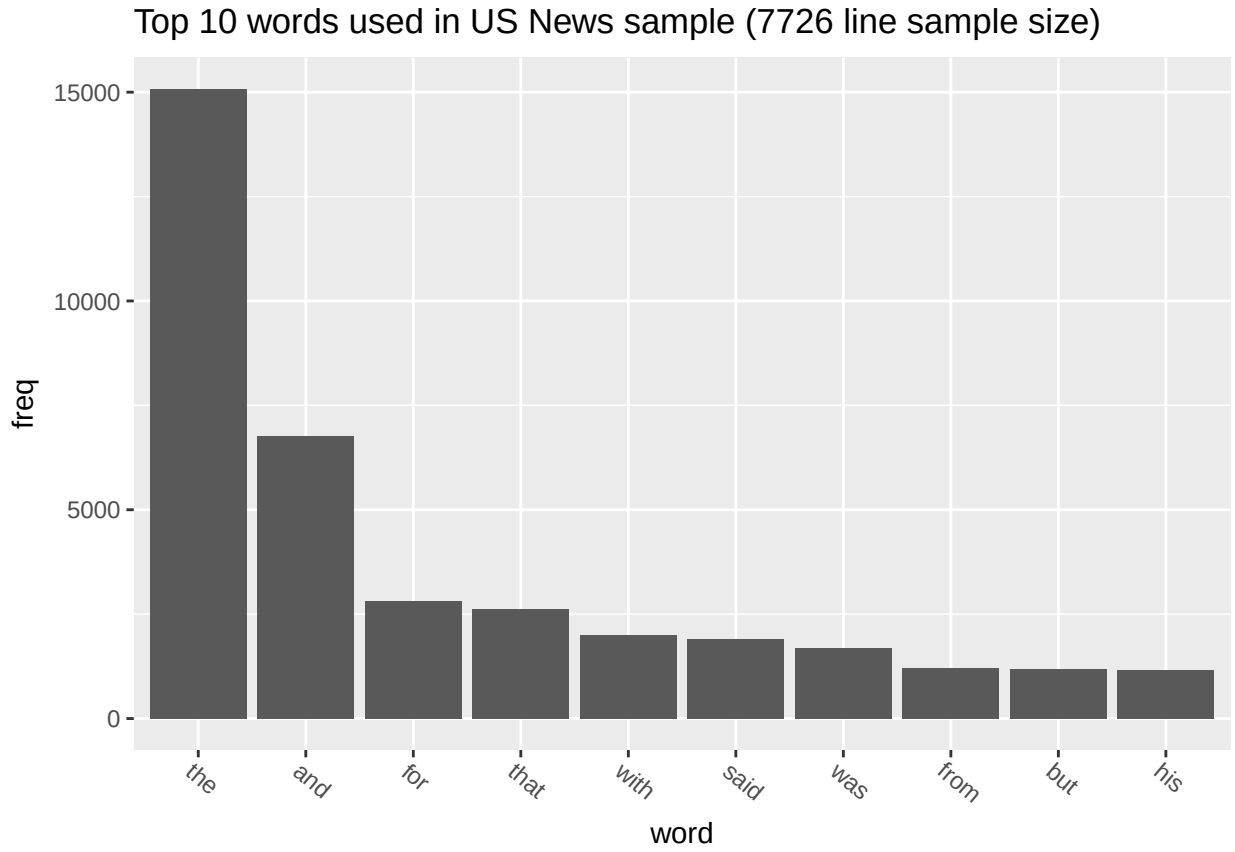


3. Cleanup data

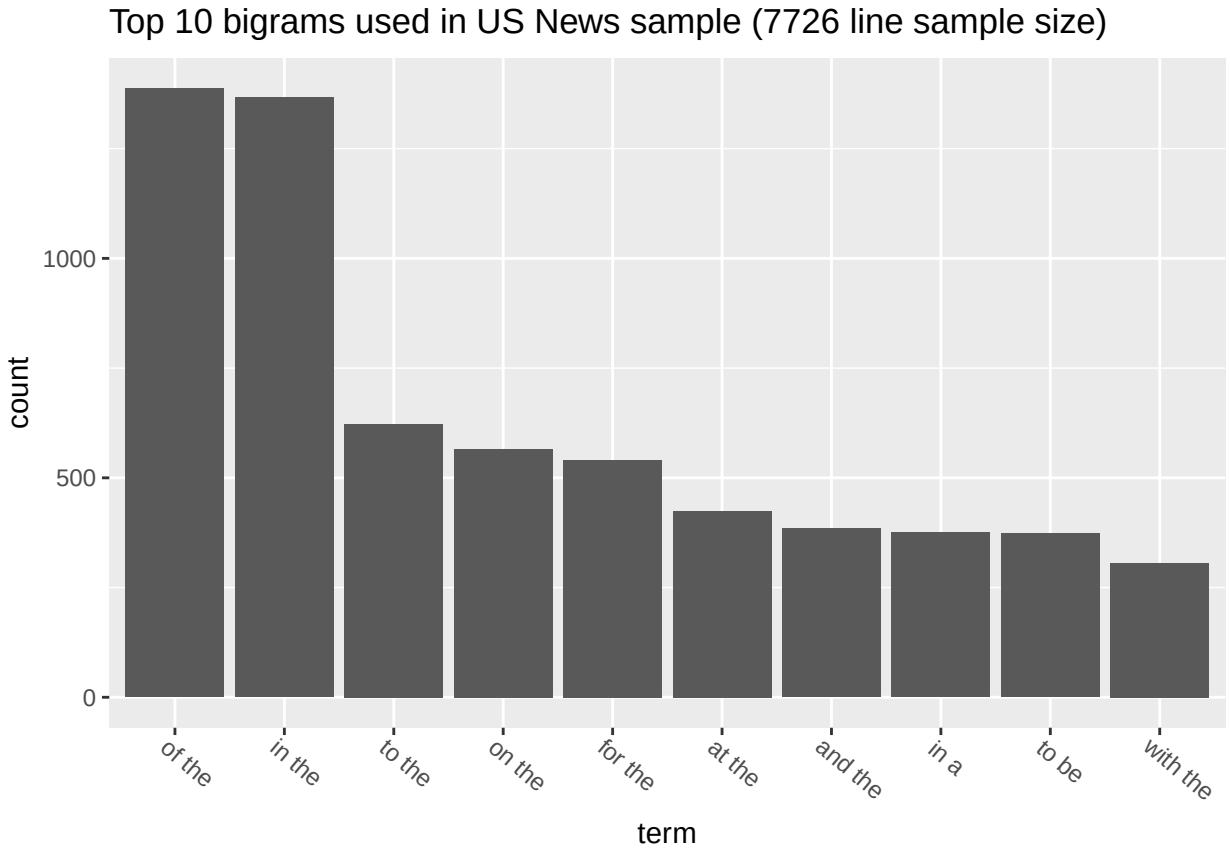
Now that we know some basic information about the data, we will want to remove things like punctuation and capital letters to allow analysis. This helps in that if something like "The" is capitalized, and there is another occurrence like "the", removing the punctuation makes it clear they are the same word. If we didn't remove capitalization, it may think those are two seperate words. Also remove punctuation to avoid confusion in those being words. Note that to do all these processes is quite intensive with memory, so we need to take smaller samples. The percentage of data taken from each will be decreased depending on the size. This was decided by trial and error to see what size data the computer could handle. If memory were not an issue, I would use a larger portion of the data.

News analysis

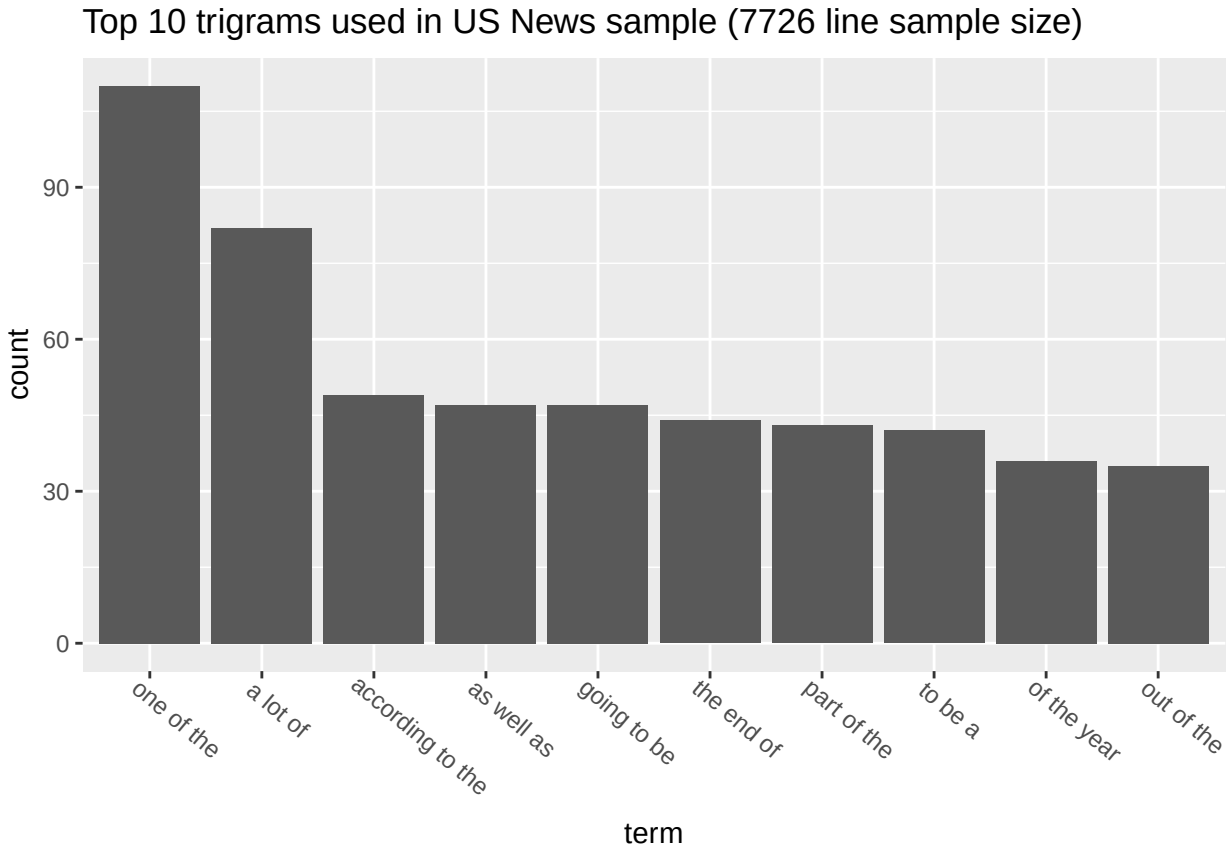
```
  # The data size are actually too big, so keep something like 10% of the data. I tried to do more, but
     NewsRecAmt = round(newsLines*0.1,digits=0)
     NewsSample = news[1:NewsRecAmt]
  #Remove punctuation
  newsNoPunct = removePunctuation(NewsSample)

  #Make all letters lowercase. It must be a list to use sapply function
  newsAllLower = sapply(list(newsNoPunct),tolower)


  #To determine word frequency, we need it to be a TermDocumentMatrix. To get to TDM, we need to first
  newsCorpus = Corpus(VectorSource(newsAllLower))
  newsTDM = TermDocumentMatrix(newsCorpus)

  #tm_map Interface to apply transformation functions (also denoted as mappings) to corpora.
  #https://stackoverflow.com/questions/15506118/make-dataframe-of-top-n-frequent-terms-for-multiple-cor


  newsTDM.matrix = as.matrix(newsTDM)
  CountWords <- sort(rowSums(newsTDM.matrix), decreasing=TRUE)

#There are a lot, try top 25 words to make plot easier to read
top10Words = CountWords[1:10]
top10 = data.frame(word=names(top10Words),freq=top10Words)
rownames(top10)= c()

#In order to get the order to not be alphabetical, need to change to character and THEN back to factor.
#https://stackoverflow.com/questions/12774210/how-do-you-specifically-order-ggplot2-x-axis-instead-of-a
top10$word = as.character(top10$word)
top10$word = factor(top10$word,levels=unique(top10$word))

ggplot(top10,aes(x=word,y=freq)) + ggtitle(paste0("Top 10 words used in US News sample (",NewsRecAmt,"
```




```
# ngrams can be calculated using term_stats in the corpus package.
#https://stackoverflow.com/questions/8898521/finding-2-3-word-phrases-using-r-tm-package
NewsBigrams = term_stats(newsCorpus,ngrams=2)
NewsTrigrams= term_stats(newsCorpus,ngrams=3)

Top10bigrams = NewsBigrams[1:10,1:2]
Top10bigrams$term = as.character(Top10bigrams$term)
Top10bigrams$term  = factor(Top10bigrams$term ,levels=unique(Top10bigrams$term))
ggplot(Top10bigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 bigrams used in US News sample (",News
```




```
Top10trigrams = NewsTrigrams[1:10,1:2]
Top10trigrams$term = as.character(Top10trigrams$term)
Top10trigrams$term  = factor(Top10trigrams$term ,levels=unique(Top10trigrams$term))
ggplot(Top10trigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 trigrams used in US News sample (",N
```



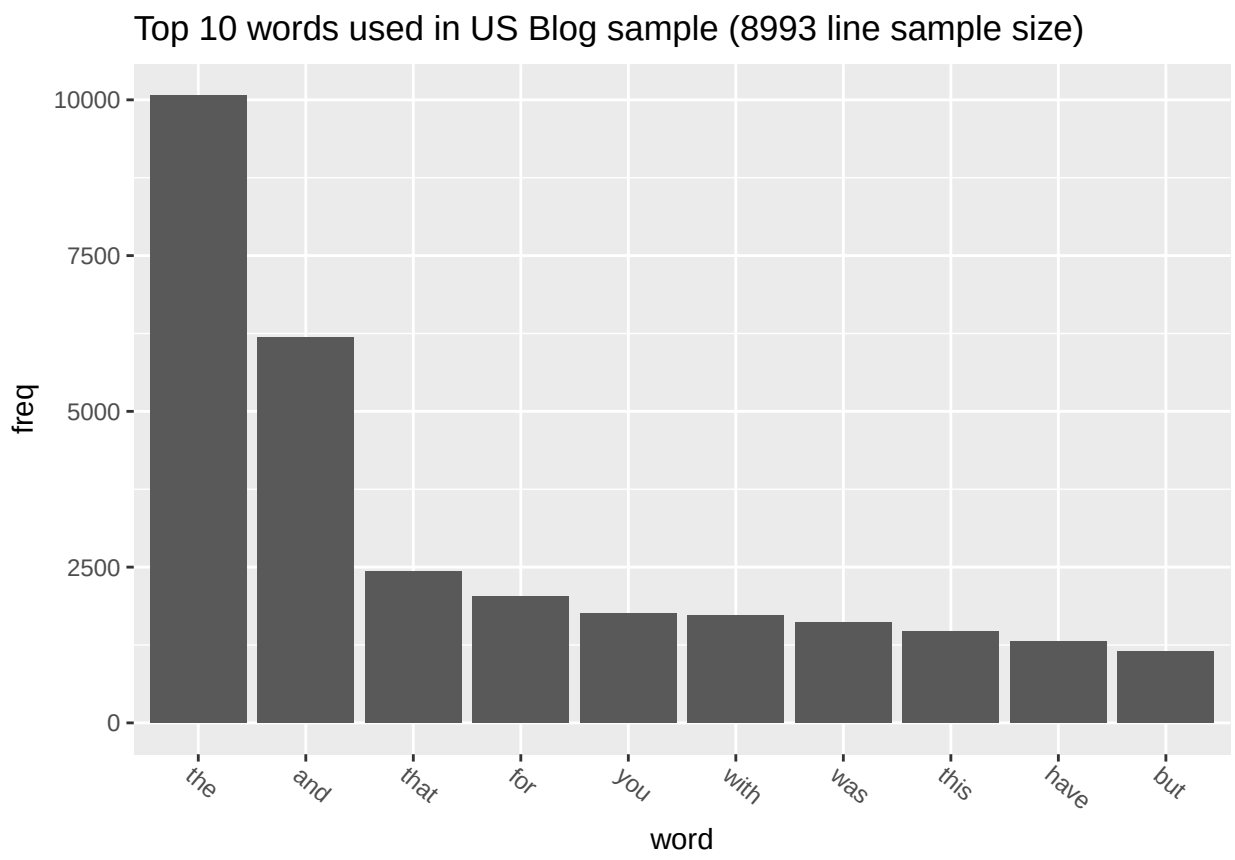

```
#Cleanup news variables to free up memory for other analysis
rm(list=ls(pattern = "news"))
rm(list=ls(pattern = "News"))
rm(list=ls(pattern="Top"))
rm(list=ls(pattern="top"))

gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  5041486 269.3    9968622  532.4   9968622  532.4
## Vcells 62121105 474.0  491437364 3749.4 520253387 3969.3
```

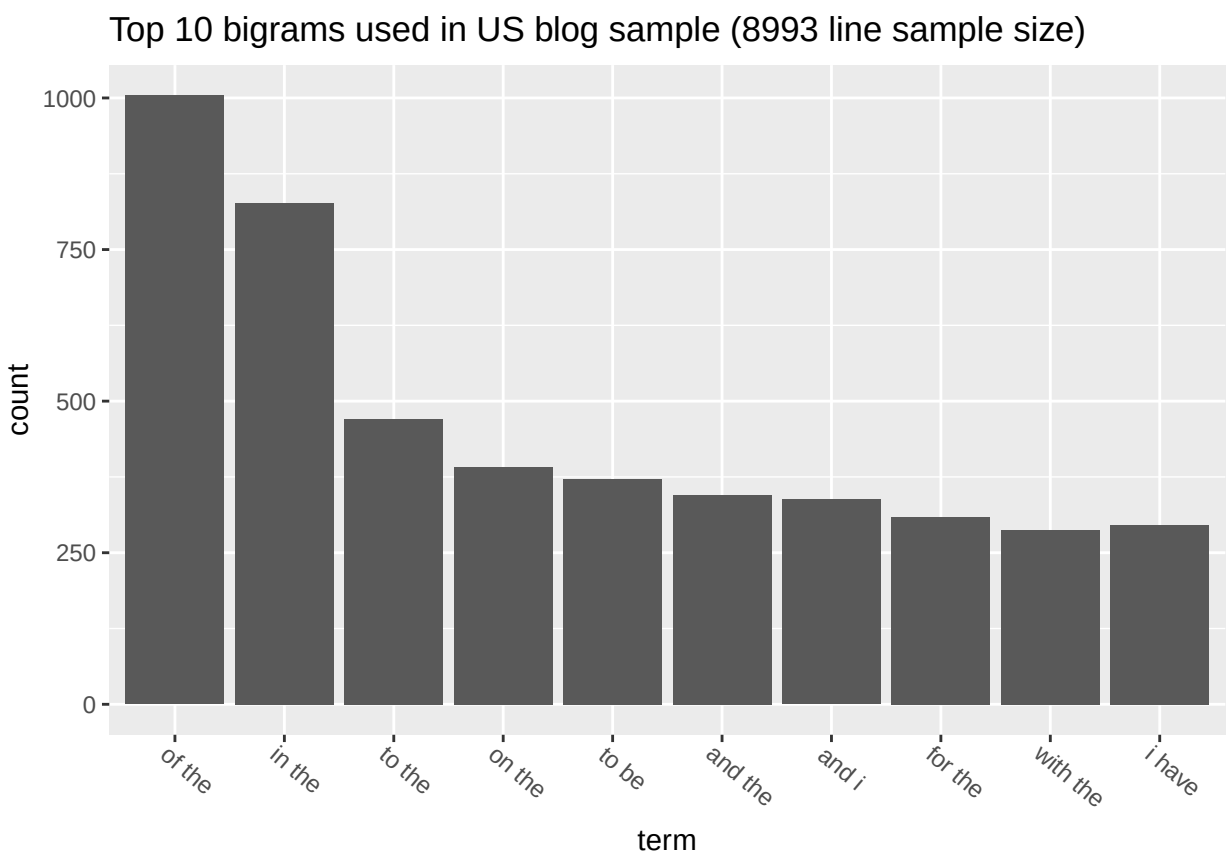
Blogs analysis

```
# The blog data based was way bigger, and proved that I had to really chop down the data to keep my com
BlogsRecAmt = round(blogLines*0.01,digits=0)
BlogsSample = blogs[1:BlogsRecAmt]
#Remove punctuation
blogsNoPunct = removePunctuation(BlogsSample)

#Issue where some of these are not ascii characters
rmNonAsci<- grep("blogsNoPunct", iconv(blogsNoPunct, "latin1", "ASCII", sub="blogsNoPunct"))
blogsFixChar = blogsNoPunct[-rmNonAsci]
```

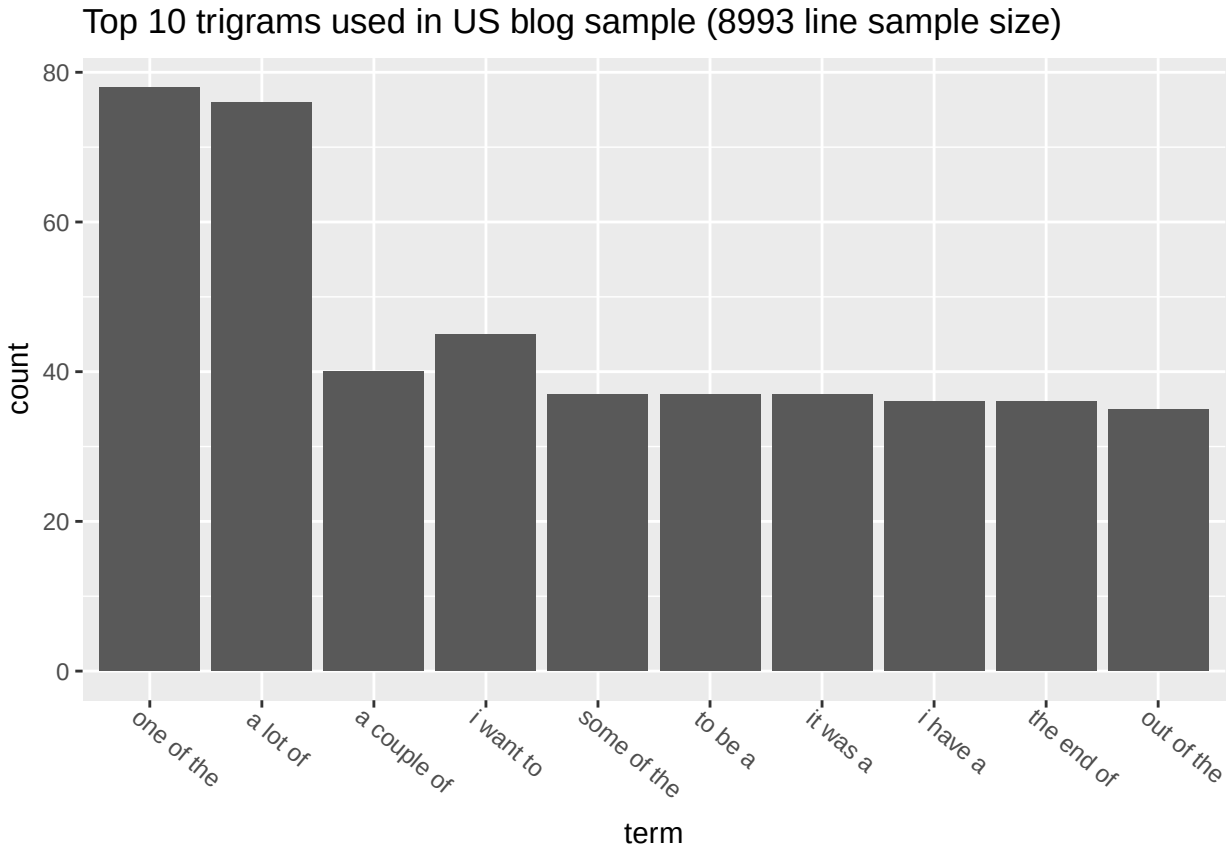
```r
#Make all letters lowercase. It must be a list to use sapply function
blogsAllLower = sapply(list(blogsFixChar),tolower)

#To determine word frequency, we need it to be a TermDocumentMatrix. To get to TDM, we need to first ge
blogsCorpus = Corpus(VectorSource(blogsAllLower))
blogsTDM = TermDocumentMatrix(blogsCorpus)

#tm_map Interface to apply transformation functions (also denoted as mappings) to corpora.
#https://stackoverflow.com/questions/15506118/make-dataframe-of-top-n-frequent-terms-for-multiple-corpo


blogsTDM.matrix = as.matrix(blogsTDM)
CountWords <- sort(rowSums(blogsTDM.matrix), decreasing=TRUE)

#There are a lot, try top 25 words to make plot easier to read
top10Words = CountWords[1:10]
top10 = data.frame(word=names(top10Words),freq=top10Words)
rownames(top10)= c()

#In order to get the order to not be alphabetical, need to change to character and THEN back to factor.
#https://stackoverflow.com/questions/12774210/how-do-you-specifically-order-ggplot2-x-axis-instead-of-a
top10$word = as.character(top10$word)
top10$word = factor(top10$word,levels=unique(top10$word))

ggplot(top10,aes(x=word,y=freq)) + ggtitle(paste0("Top 10 words used in US Blog sample (",BlogsRecAmt,"
```



```
# ngrams can be calculated using term_stats in the corpus package.
#https://stackoverflow.com/questions/8898521/finding-2-3-word-phrases-using-r-tm-package
blogsBigrams = term_stats(blogsCorpus,ngrams=2)
blogsTrigrams= term_stats(blogsCorpus,ngrams=3)

Top10bigrams = blogsBigrams[1:10,1:2]
Top10bigrams$term = as.character(Top10bigrams$term)
Top10bigrams$term  = factor(Top10bigrams$term ,levels=unique(Top10bigrams$term))
ggplot(Top10bigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 bigrams used in US blog sample (",Blo
```




```
Top10trigrams = blogsTrigrams[1:10,1:2]
Top10trigrams$term = as.character(Top10trigrams$term)
Top10trigrams$term  = factor(Top10trigrams$term ,levels=unique(Top10trigrams$term))
ggplot(Top10trigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 trigrams used in US blog sample (",B
```



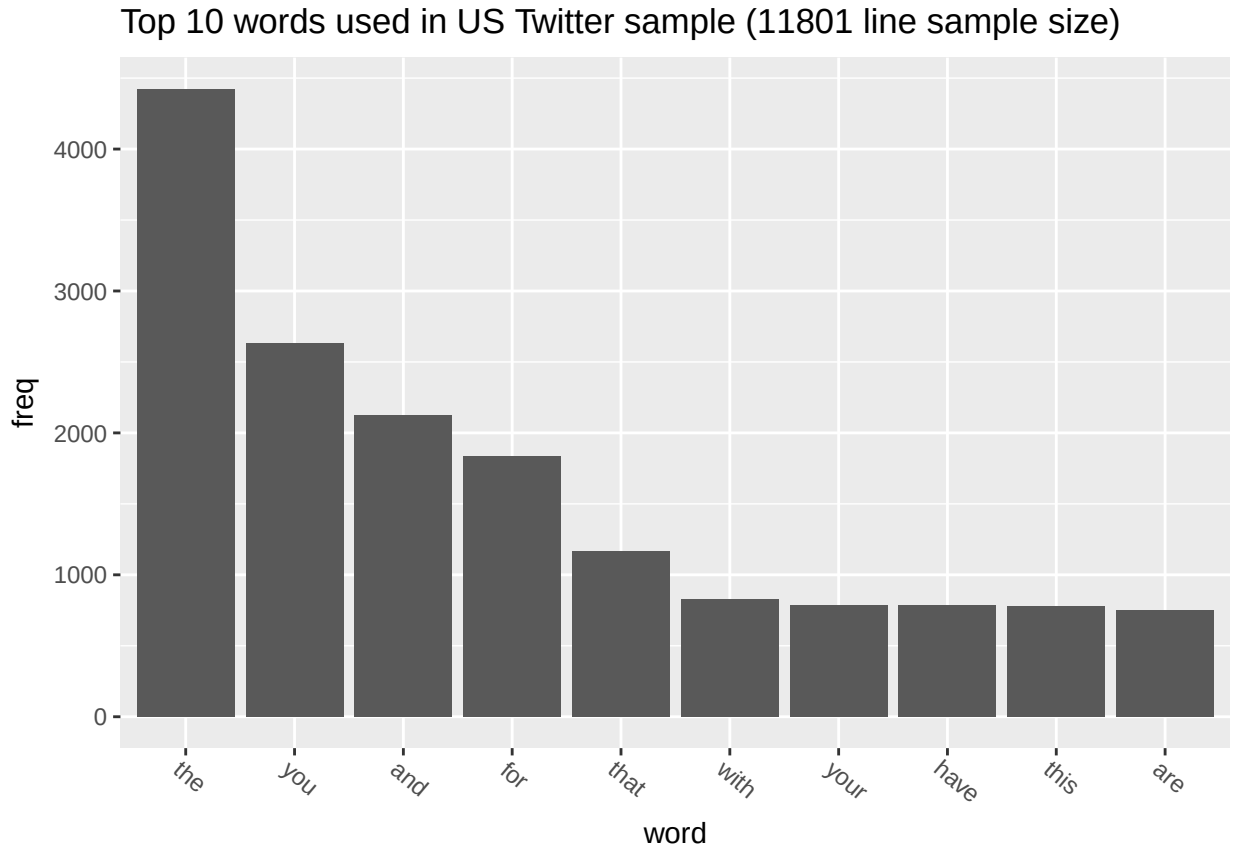

```
#Clean up blog variables to make room for twitter analysis
rm(list=ls(pattern = "blogs"))
rm(list=ls(pattern = "Blogs"))
rm(list=ls(pattern = "blog"))
rm(list=ls(pattern="Top"))
rm(list=ls(pattern="top"))
rm(CountWords)
rm(rmNonAsci)
gc()
```

```
##            used  (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells  4117136 219.9    9968622  532.4   9968622  532.4
## Vcells 33794549 257.9  393149891 2999.5 520253387 3969.3
```

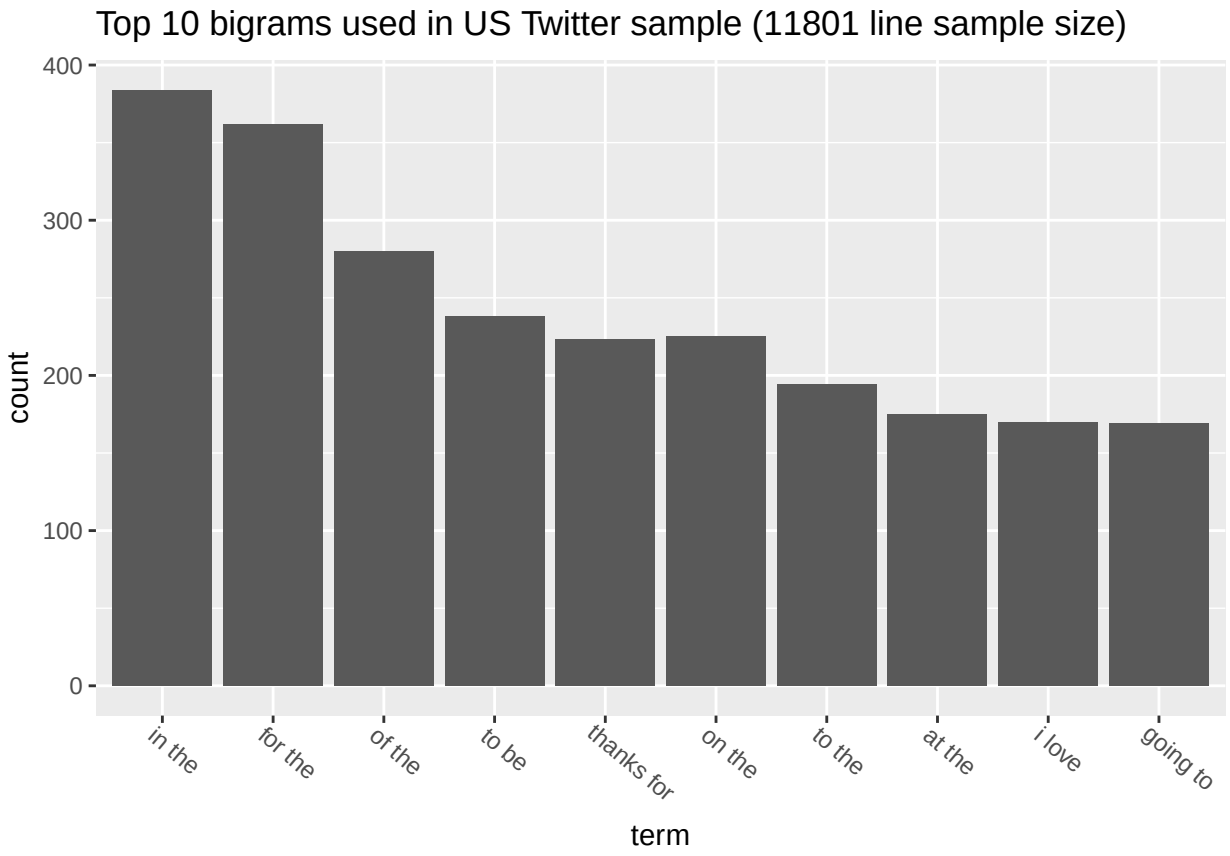
Twitter analysis

```
# The data size are actually too big, so keep something like 10% of the data as memory has been an issu
TwitterRecAmt = round(tweetLines*0.005,digits=0)
TwitterSample = tweets[1:TwitterRecAmt]
#Remove punctuation
twitterNoPunct = removePunctuation(TwitterSample)

#Issue where some of these are not ascii characters
rmNonAsci<- grep("twitterNoPunct", iconv(twitterNoPunct, "latin1", "ASCII", sub="twitterNoPunct"))
```

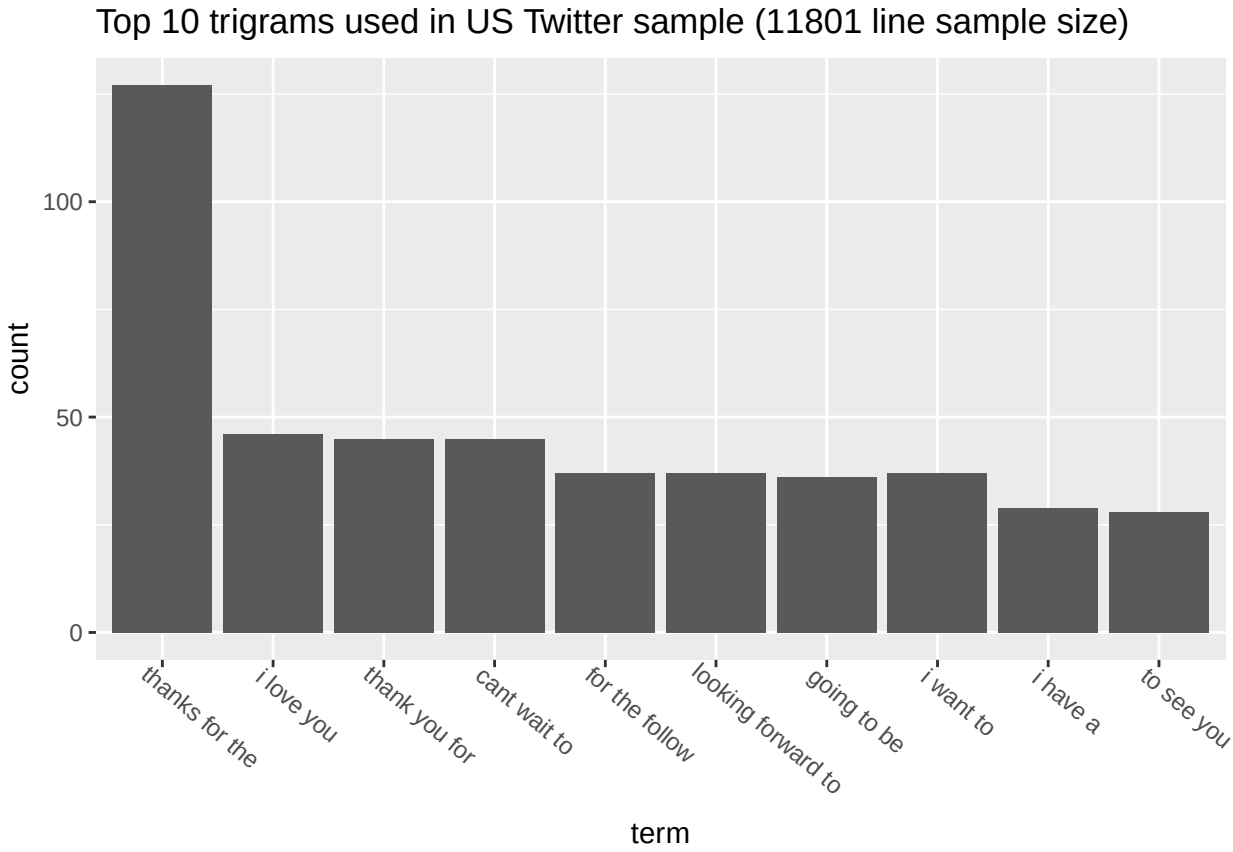
```
twitterFixChar = twitterNoPunct[-rmNonAsci]

#Make all letters lowercase. It must be a list to use sapply function
twitterAllLower = sapply(list(twitterFixChar),tolower)

#To determine word frequency, we need it to be a TermDocumentMatrix. To get to TDM, we need to first ge
twitterCorpus = Corpus(VectorSource(twitterAllLower))
twitterTDM = TermDocumentMatrix(twitterCorpus)

#tm_map Interface to apply transformation functions (also denoted as mappings) to corpora.
#https://stackoverflow.com/questions/15506118/make-dataframe-of-top-n-frequent-terms-for-multiple-corpo


twitterTDM.matrix = as.matrix(twitterTDM)
CountWords <- sort(rowSums(twitterTDM.matrix), decreasing=TRUE)

#There are a lot, try top 25 words to make plot easier to read
top10Words = CountWords[1:10]
top10 = data.frame(word=names(top10Words),freq=top10Words)
rownames(top10)= c()

#In order to get the order to not be alphabetical, need to change to character and THEN back to factor.
#https://stackoverflow.com/questions/12774210/how-do-you-specifically-order-ggplot2-x-axis-instead-of-a
top10$word = as.character(top10$word)
top10$word = factor(top10$word,levels=unique(top10$word))

ggplot(top10,aes(x=word,y=freq)) + ggtitle(paste0("Top 10 words used in US Twitter sample (",TwitterRec
```




```
# ngrams can be calculated using term_stats in the corpus package.
#https://stackoverflow.com/questions/8898521/finding-2-3-word-phrases-using-r-tm-package
twitterBigrams = term_stats(twitterCorpus,ngrams=2)
twitterTrigrams= term_stats(twitterCorpus,ngrams=3)

Top10bigrams = twitterBigrams[1:10,1:2]
Top10bigrams$term = as.character(Top10bigrams$term)
Top10bigrams$term  = factor(Top10bigrams$term ,levels=unique(Top10bigrams$term))
ggplot(Top10bigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 bigrams used in US Twitter sample (",
```




```
Top10trigrams = twitterTrigrams[1:10,1:2]
Top10trigrams$term = as.character(Top10trigrams$term)
Top10trigrams$term  = factor(Top10trigrams$term ,levels=unique(Top10trigrams$term))
ggplot(Top10trigrams,aes(x=term,y=count)) + ggtitle(paste0("Top 10 trigrams used in US Twitter sample (
```


Top 10 trigrams used in US Twitter sample (11801 line sample size)
count
100
50
0
thanks for the
i love you
thank you for
cant wait to
for the follow
looking forward to
going to be
i want to
i have a
to see you
term