Text Data Analysis Report - Week 2 - Task

Berj Dekramanjian

2024-06-07

Introduction

This report presents an analysis of text data from three different sources: blogs, Twitter, and news articles. The analysis includes data summarization, cleaning, tokenization, and visualization of unigram, bigram, and trigram frequencies. Additionally, a sentiment analysis is performed on the data.

Load Necessary Libraries

We load the required R libraries like tidyverse, tm, and tidytext, which are essential for data processing, text analysis, and visualization.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(shiny)
library(knitr)
library(stringr)
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(tidytext)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(textdata)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

Load and Inspect the Data

We use the readLines function to extract data from the text files and assign them to three variables (blogs, twitter, and news). We then use the head function to check the top lines of each dataset to understand its structure.

```
# Load the data
blogs <- readLines("en_US.blogs.txt", warn = FALSE, encoding = "UTF-8")
twitter <- readLines("en_US.twitter.txt", warn = FALSE, encoding = "UTF-8")
news <- readLines("en_US.news.txt", warn = FALSE, encoding = "UTF-8")

# Check the data
head(blogs)
```

```
## [1] "In the years thereafter, most of the Oil fields and platforms were named after pagan "gods"."
## [2] "We love you Mr. Brown."
## [3] "Chad has been awesome with the kids and holding down the fort while I work later than usual! Th
## [4] "so anyways, i am going to share some home decor inspiration that i have been storing in my fold
## [5] "With graduation season right around the corner, Nancy has whipped up a fun set to help you out
## [6] "If you have an alternative argument, let's hear it! :)"
```

```
head(twitter)
```

```
## [1] "How are you? Btw thanks for the RT. You gonna be in DC anytime soon? Love to see you. Been way,
## [2] "When you meet someone special... you'll know. Your heart will beat more rapidly and you'll smil
## [3] "they've decided its more fun if I don't."
## [4] "So Tired D; Played Lazer Tag & Ran A LOT D; Ughh Going To Sleep Like In 5 Minutes ;)"
## [5] "Words from a complete stranger! Made my birthday even better :)"
## [6] "First Cubs game ever! Wrigley field is gorgeous. This is perfect. Go Cubs Go!"
```

```
head(news)
```

```
## [1] "He wasn't home alone, apparently."
## [2] "The St. Louis plant had to close. It would die of old age. Workers had been making cars there s
## [3] "WSU's plans quickly became a hot topic on local online sites. Though most people applauded plans
## [4] "The Alaimo Group of Mount Holly was up for a contract last fall to evaluate and suggest improve
## [5] "And when it's often difficult to predict a law's impact, legislators should think twice before
## [6] "There was a certain amount of scoffing going around a few years ago when the NFL decided to mov
```

Summary Statistics

We create summaries for each dataset by calculating the number of characters, lines, and words, and the file size. We combine these summaries into a single table for easy comparison.

```
# Create summaries of each
blogs_summary <- tibble(
  File = "Blogs",
  Characters = sum(nchar(blogs)),
  Lines = length(blogs),
  Words = sum(str_count(blogs, "\\w+")),
  Size_MB = file.size("en_US.blogs.txt") / (1024^2)
)

twitter_summary <- tibble(
  File = "Twitter",
  Characters = sum(nchar(twitter)),
  Lines = length(twitter),
  Words = sum(str_count(twitter, "\\w+")),
  Size_MB = file.size("en_US.twitter.txt") / (1024^2)
)

news_summary <- tibble(
  File = "News",
  Characters = sum(nchar(news)),
  Lines = length(news),
  Words = sum(str_count(news, "\\w+")),
  Size_MB = file.size("en_US.news.txt") / (1024^2)
)

# Combine all summaries into one table
summary_stats <- bind_rows(blogs_summary, twitter_summary, news_summary)
kable(summary_stats, caption = "Summary Statistics of Text Data")
```

Table 1: Summary Statistics of Text Data

File	Characters	Lines	Words	Size_MB
Blogs	206824505	899288	38309620	200.4242
Twitter	162096031	2360148	31003501	159.3641
News	15639408	77259	2741594	196.2775

Data Sampling and Cleaning

We sample 5000 lines from each dataset to create a manageable subset for analysis. We then create a text corpus and clean it by converting the text to lowercase, removing punctuation, numbers, extra whitespace, stopwords, and profanities. We also remove URLs.

```
# Sample the data to create manageable datasets for plotting, choosing 1000 lines from each
set.seed(123)
blogs_sample <- sample(blogs, 5000)
twitter_sample <- sample(twitter, 5000)
```

```r
news_sample <- sample(news, 5000)

text_samples <- c(blogs_sample, twitter_sample, news_sample)

# Create a corpus
corpus <- Corpus(VectorSource(text_samples))

# Clean the corpus
# Load a list of profanities in English
profanity <- read.csv("profanity_en.csv", header = FALSE, stringsAsFactors = FALSE)
profanity <- profanity$V1

corpus_clean <- tm_map(corpus, content_transformer(tolower)) # Convert text to lower case
corpus_clean <- tm_map(corpus_clean, removePunctuation) # Remove punctuations
corpus_clean <- tm_map(corpus_clean, removeNumbers) # Remove numbers
corpus_clean <- tm_map(corpus_clean, stripWhitespace) # Remove extra whitespaces
corpus_clean <- tm_map(corpus_clean, removeWords, stopwords("en")) # Remove common stopwords
corpus_clean <- tm_map(corpus_clean, removeWords, profanity) # Remove profanities

# Remove URLs
remove_url <- content_transformer(function(x) gsub("(f|ht)tp(s?)://(.*)[.][a-z]+", "", x))
corpus_clean <- tm_map(corpus_clean, remove_url)

# Convert cleaned corpus to a data frame
corpus_df <- data.frame(text = sapply(corpus_clean, as.character), stringsAsFactors = FALSE)
```

Tokenization and Frequency Analysis

Unigrams

We tokenize the cleaned text data into individual words (unigrams) and calculate their frequencies. We display the top 10 most frequent unigrams and create a bar chart for the top 30 unigrams.

```r
# Tokenize into unigrams
unigrams <- corpus_df %>%
  unnest_tokens(word, text)

# Calculate unigram frequencies
unigram_freq <- unigrams %>%
  count(word, sort = TRUE)

# Display the top 10 unigrams
print("Top 10 Unigrams")
```

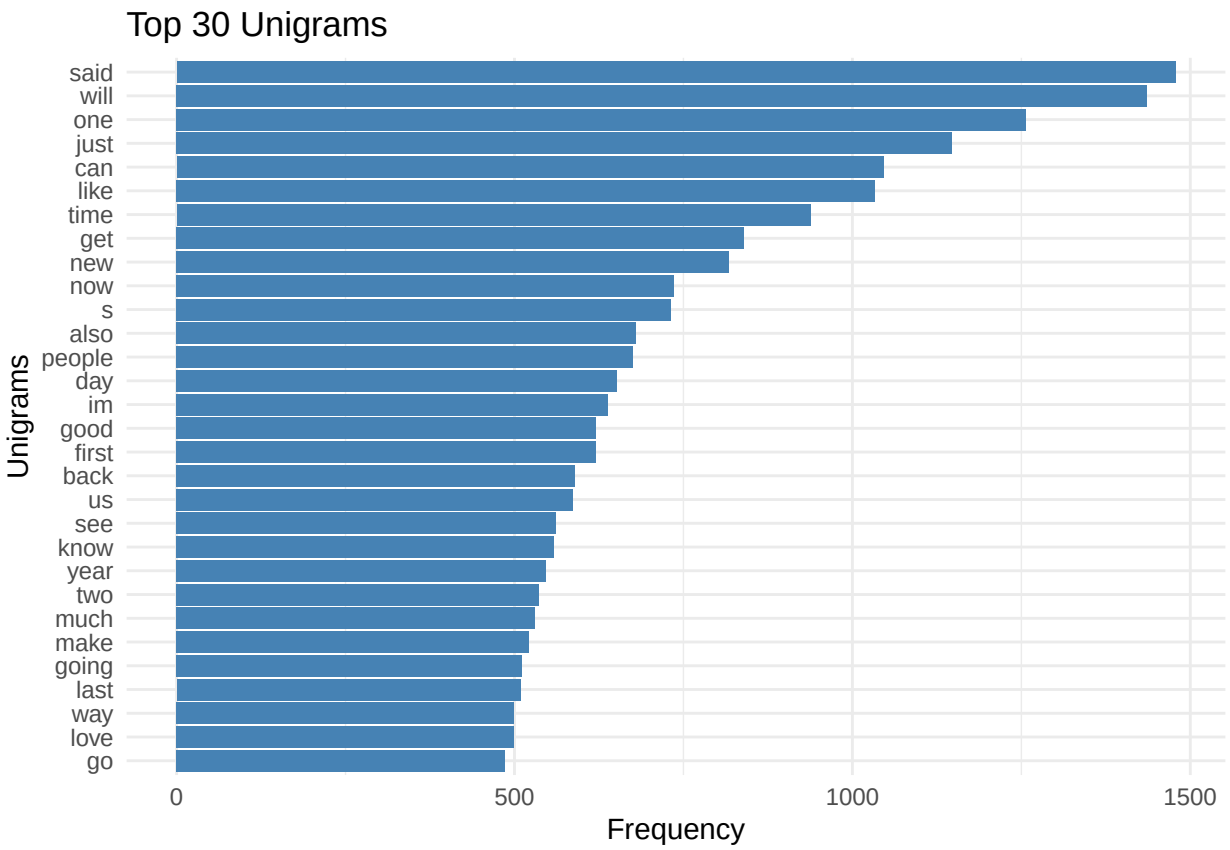
```
## [1] "Top 10 Unigrams"
```

```r
print(head(unigram_freq, 10))
```

```
##    word    n
## 1  said 1478
## 2  will 1436
```

```
## 3   one 1257
## 4  just 1147
## 5   can 1046
## 6  like 1034
## 7  time  938
## 8   get  840
## 9   new  818
## 10  now  736
```

```r
top30_unigrams <- unigram_freq %>% top_n(30, wt = n)
ggplot(top30_unigrams, aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 30 Unigrams", x = "Unigrams", y = "Frequency") +
  theme_minimal()
```



Bigrams

We similarly tokenize the text data into pairs of consecutive words (bigrams) and calculate their frequencies. We display the top 10 bigrams and create a bar chart for the top 30 bigrams.

```r
# Tokenize into bigrams
bigrams <- corpus_df %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  filter(!is.na(bigram))  # Remove NA values
```

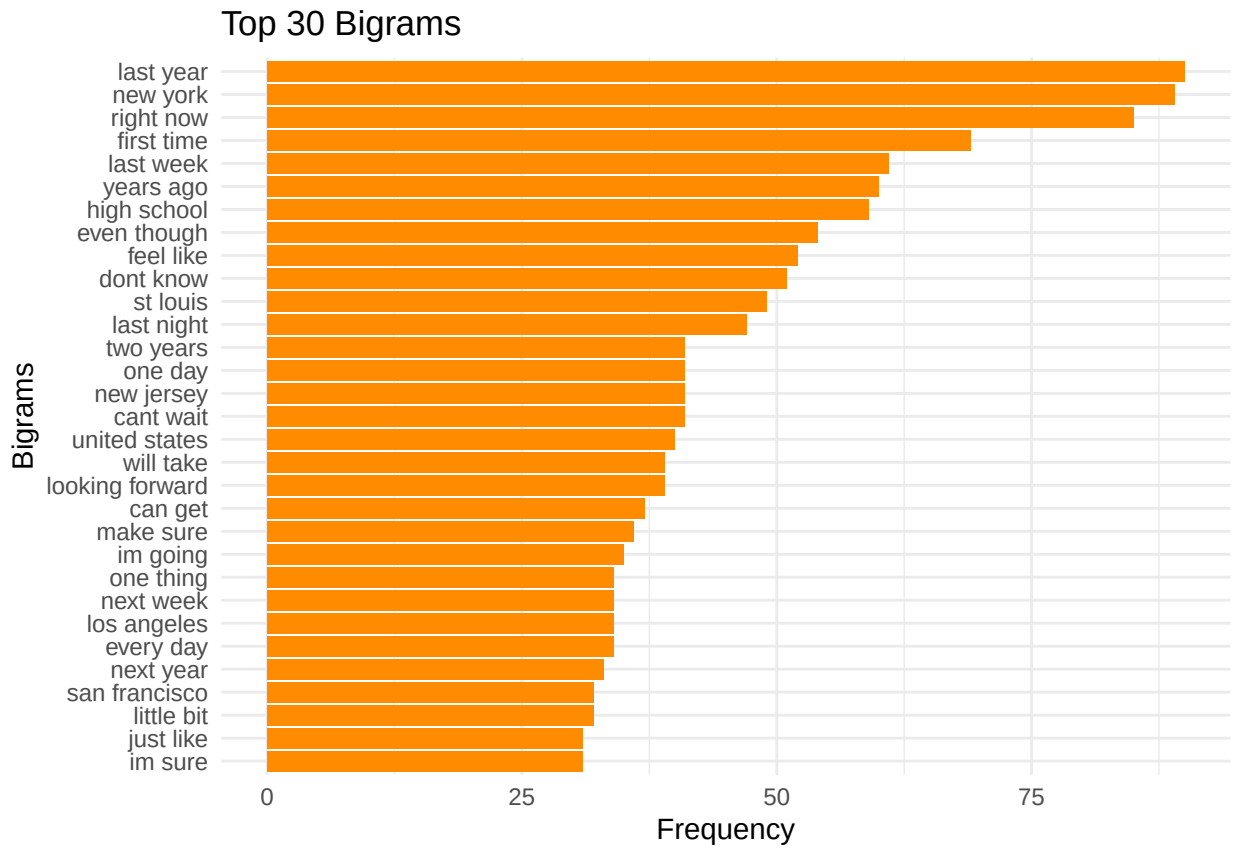
```r
# Calculate bigram frequencies
bigram_freq <- bigrams %>%
  count(bigram, sort = TRUE)

# Display the top 10 bigrams
print("Top 10 Bigrams")
```

```
## [1] "Top 10 Bigrams"
```

```r
print(head(bigram_freq, 10))
```

```
##         bigram  n
## 1    last year 90
## 2     new york 89
## 3    right now 85
## 4   first time 69
## 5    last week 61
## 6    years ago 60
## 7  high school 59
## 8  even though 54
## 9    feel like 52
## 10   dont know 51
```

```r
top30_bigrams <- bigram_freq %>% top_n(30, wt = n)

ggplot(top30_bigrams, aes(x = reorder(bigram, n), y = n)) +
  geom_bar(stat = "identity", fill = "darkorange") +
  coord_flip() +
  labs(title = "Top 30 Bigrams", x = "Bigrams", y = "Frequency") +
  theme_minimal()
```



Trigrams

We lastly tokenize the text data into triplets of consecutive words (trigrams) and calculate their frequencies. We display the top 10 trigrams and create a bar chart for the top 20 trigrams.

```
# Tokenize into trigrams
trigrams <- corpus_df %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  filter(!is.na(trigram))  # Remove NA values

# Calculate trigram frequencies
trigram_freq <- trigrams %>%
  count(trigram, sort = TRUE)

# Display the top 10 trigrams
print("Top 10 Trigrams")
```

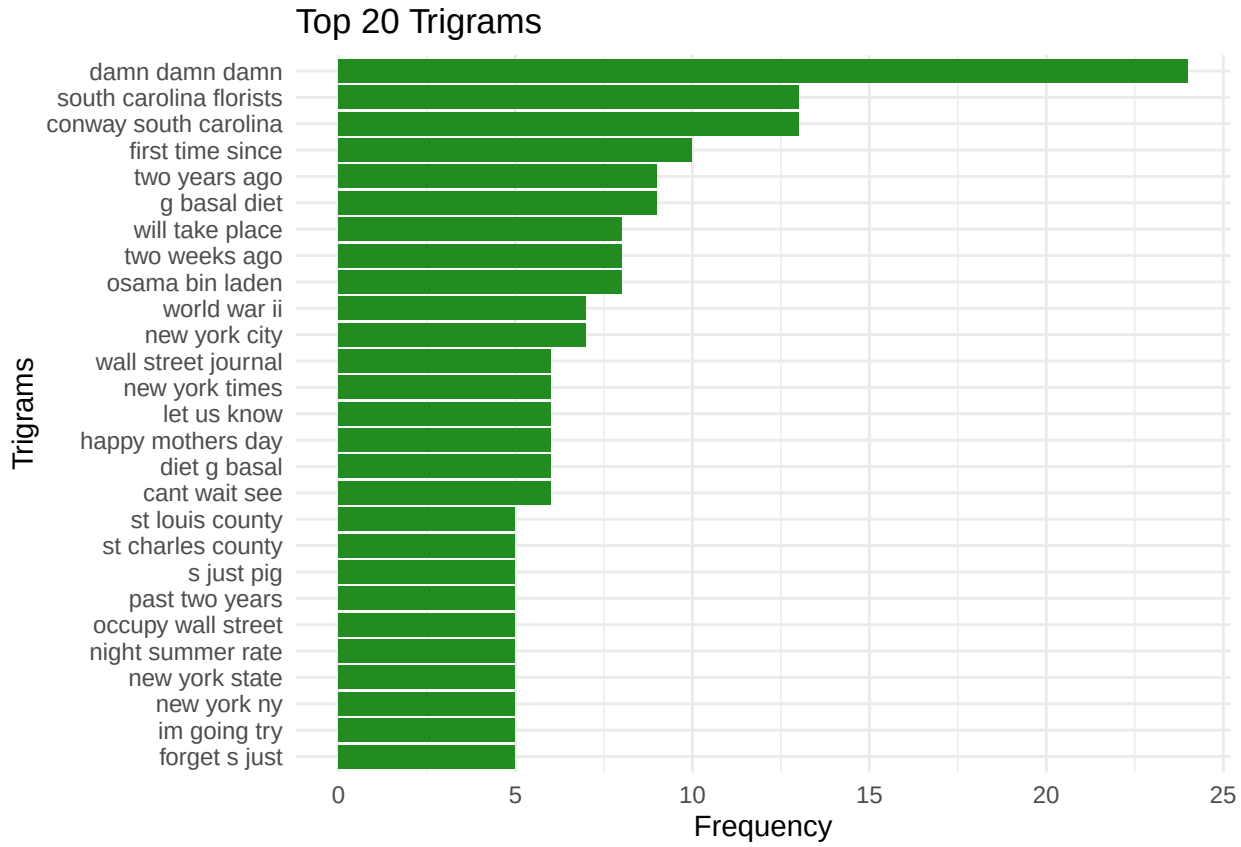
```
## [1] "Top 10 Trigrams"
```

```
print(head(trigram_freq, 10))
```

```
##                    trigram  n
## 1           damn damn damn 24
## 2    conway south carolina 13
```

```
## 3  south carolina florists 13
## 4         first time since 10
## 5             g basal diet  9
## 6            two years ago  9
## 7          osama bin laden  8
## 8            two weeks ago  8
## 9          will take place  8
## 10           new york city  7
```

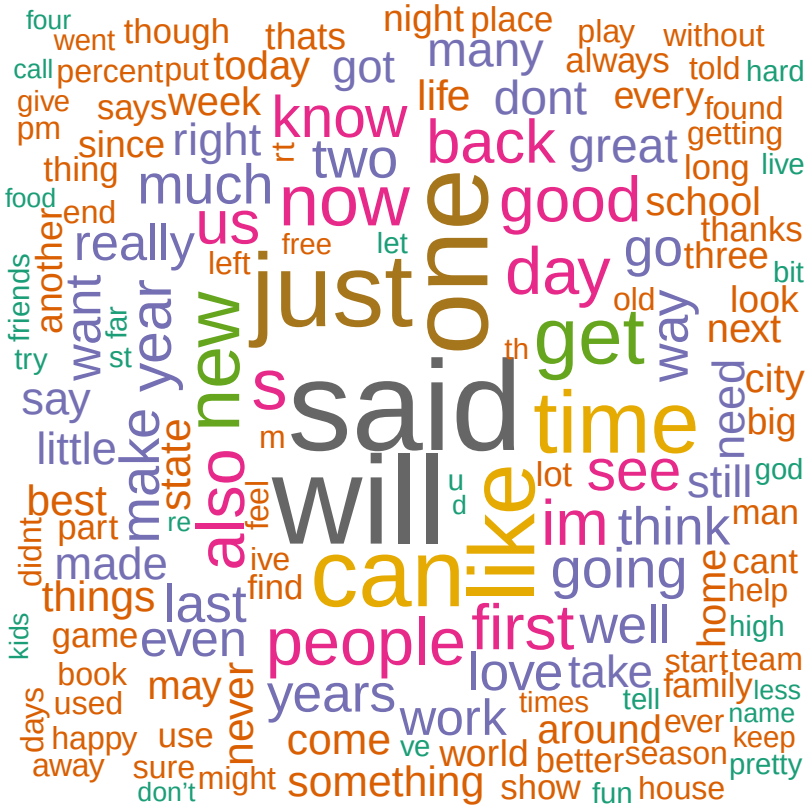
```r
top20_trigrams <- trigram_freq %>% top_n(20, wt = n)

ggplot(top20_trigrams, aes(x = reorder(trigram, n), y = n)) +
  geom_bar(stat = "identity", fill = "forestgreen") +
  coord_flip() +
  labs(title = "Top 20 Trigrams", x = "Trigrams", y = "Frequency") +
  theme_minimal()
```



Word Cloud

We create a word cloud to visually represent the frequency of unigrams in the text data, providing a clear visual of the most common words.

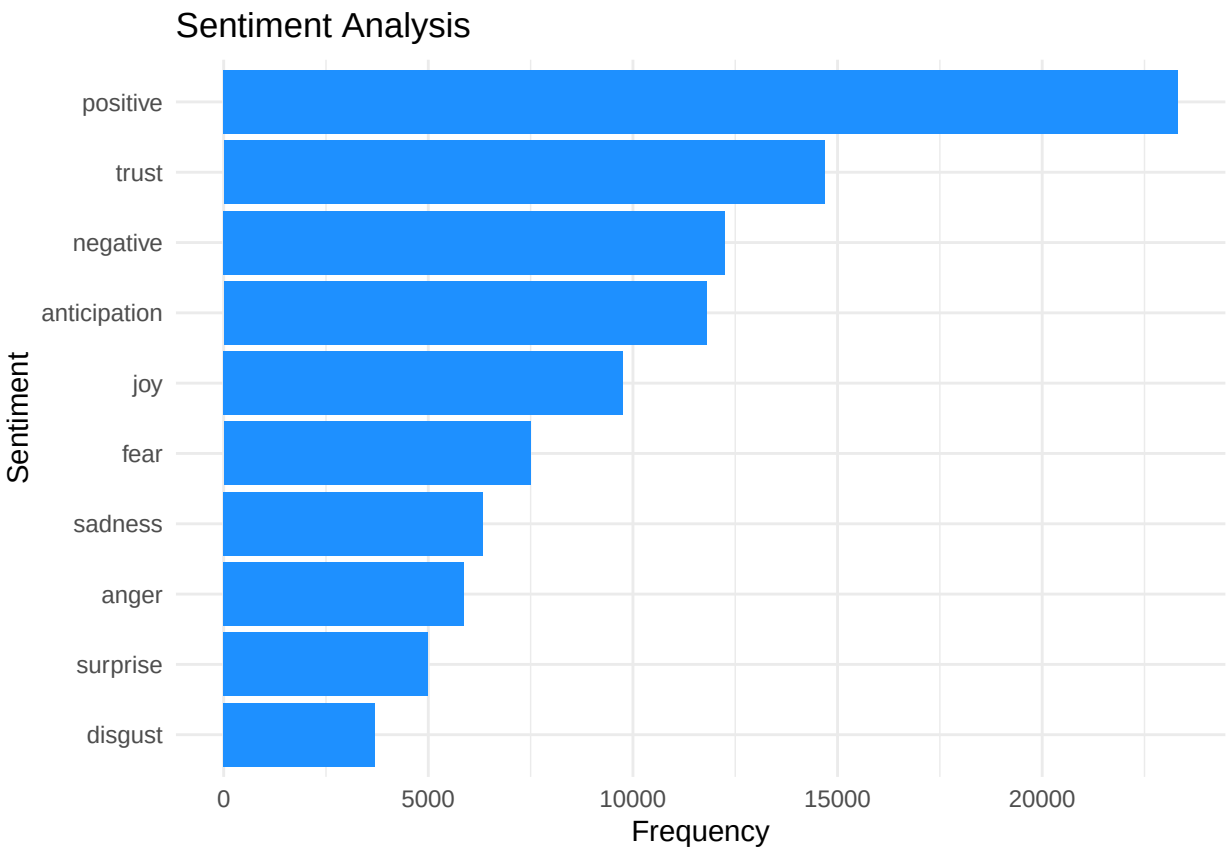
```r
set.seed(1234)
wordcloud(words = unigram_freq$word, freq = unigram_freq$n, min.freq = 50,
          max.words = 200, random.order = FALSE, colors = brewer.pal(8, "Dark2"))
```



Sentiment Analysis

We perform sentiment analysis using the NRC sentiment lexicon to determine the sentiments expressed in the text data. We then present the frequency of each sentiment in a bar chart.

```
# Sentiment Analysis
nrc_sentiments <- get_sentiments("nrc")

sentiment_scores <- unigrams %>%
  inner_join(nrc_sentiments, by = "word") %>%
  count(sentiment, sort = TRUE)

ggplot(sentiment_scores, aes(x = reorder(sentiment, n), y = n)) +
  geom_bar(stat = "identity", fill = "dodgerblue") +
  coord_flip() +
  labs(title = "Sentiment Analysis", x = "Sentiment", y = "Frequency") +
  theme_minimal()
```



Conclusion

Three different text data sources are analyzed in this assignment: blogs, news and twitter. Together these sources represent over 3 million sentences containing in total over 70 million words.

The text data is sampled for efficient data handling. Next the data is cleaned (tokenization, profanity filtering) and further analysis is performed on the frequency of the words.

As one might expect, the most frequent words are often the simplest words, and its good to know the sentiment is mostly positive and trusting

Future Strategies for Shiny App

To enhance Corpus cleansing improvements. There are many still avenues to remove any anamoly/abnormal characters and words.

To increase the size of corpus for shiny application. However, performance consideration must also be factor in.

Deploying "ngram" algorithim to calculate the probabilities of the next word occuring. An inclusion of four/quad gram may be taken into consideration

Implement a backoff mechanism into shiny application

Shiny App: To create a simplier user interface with some guidelines, enabling the user to key in the text, and detect the next word leveraging on the n-gram algorithim prediction.