Milestone Report

Ruoshi Li

5/3/2020

Introduction

This milestone report is for Coursera Data Capstone course. The goal of this report is to build a predictive algorithm of text mining. More specifically, a n-gram model will be used to predict the next word based on the previous words in a text corpus. Natural language processing techiniques will be applied in the analysis.

Download Data and Data Exploration

```
# Set 'working directory'
wdir <- "/Users/alanlou/Desktop/R_Projects"
setwd(wdir)
# load necessary libraries
library(ggplot2)
library(stringi)
library(rJava)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(SnowballC)
library(RWeka)
```

Getting Data

Download the data from: https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip. The data is from three different resources: blogs, news and twitter.

```
# read data
blogs <- readLines("dataset/en_US/en_US.blogs.txt", encoding = "UTF-8", skipNul = TRUE)
news <- readLines("dataset/en_US/en_US.news.txt", encoding = "UTF-8", skipNul = TRUE)
twitter <- readLines("dataset/en_US/en_US.twitter.txt", encoding = "UTF-8", skipNul = TRUE)
# get data summary
dataset <- data.frame(File = c("blog", "news", "twitter"),
                      t(rbind(sapply(list(blogs, news, twitter), stri_stats_general),
                              TotalWords = sapply(list(blogs, news, twitter), stri_stats_latex)[4,]))
```

```
                     )
dataset
```

```
##      File   Lines LinesNEmpty     Chars CharsNWhite TotalWords
## 1    blog  899288      899288 206824382   170389539   37570839
## 2    news 1010242     1010242 203223154   169860866   34494539
## 3 twitter 2360148     2360148 162096241   134082806   30451170
```

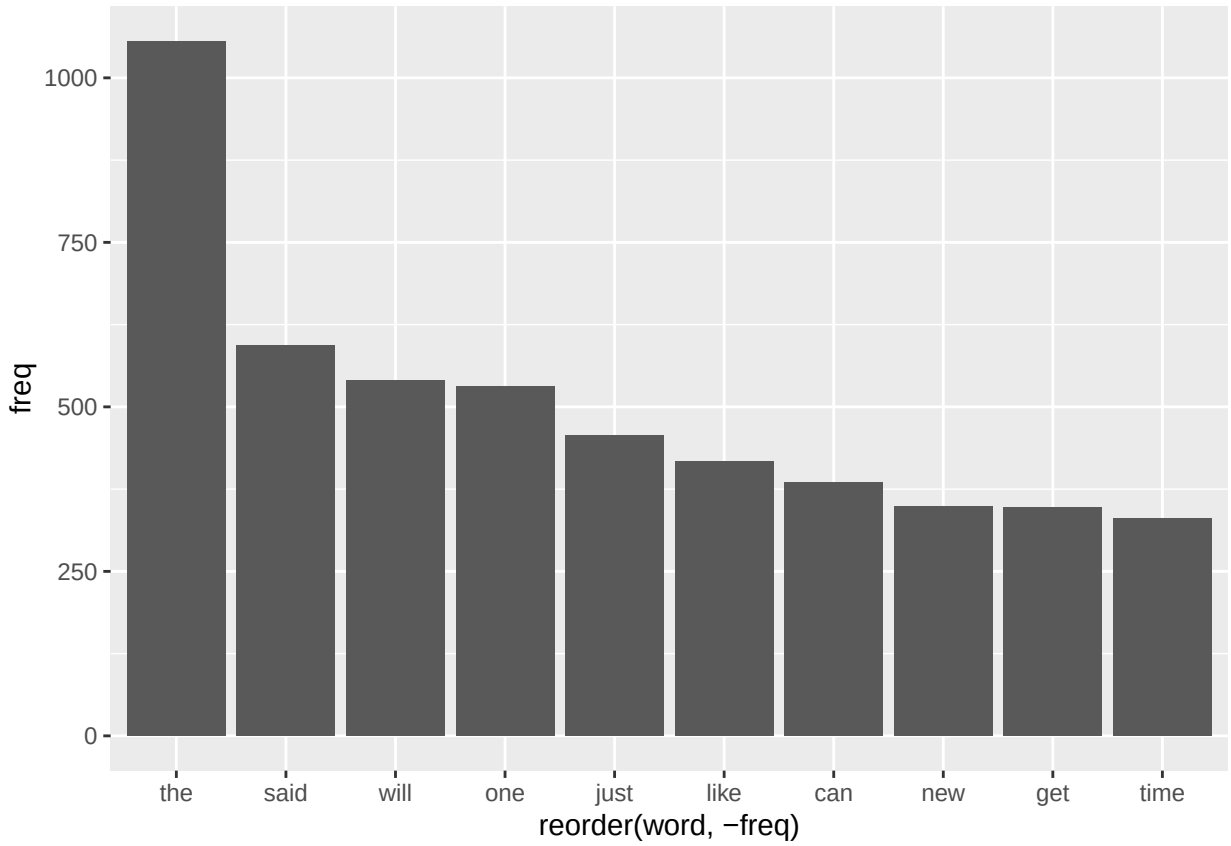
Data Sampling and Data Cleaning

```
# sampling data
set.seed(666)
blogs_sample <- blogs[sample(1: length(blogs), 2000)]
news_sample <- news[sample(1: length(news), 2000)]
twitter_sample <- twitter[sample(1: length(twitter), 2000)]
data_sample <- c(blogs_sample, news_sample, twitter_sample)
# representing and computing on corpora
data <- VCorpus(VectorSource(data_sample))
# remove spaces and special characters
data <- tm_map(data, stripWhitespace)
data <- tm_map(data, removePunctuation)
data <- tm_map(data, removeNumbers)
data <- tm_map(data, removeWords, stopwords("english"))
# turn all words to lowercase
data <- tm_map(data, content_transformer(tolower))
```

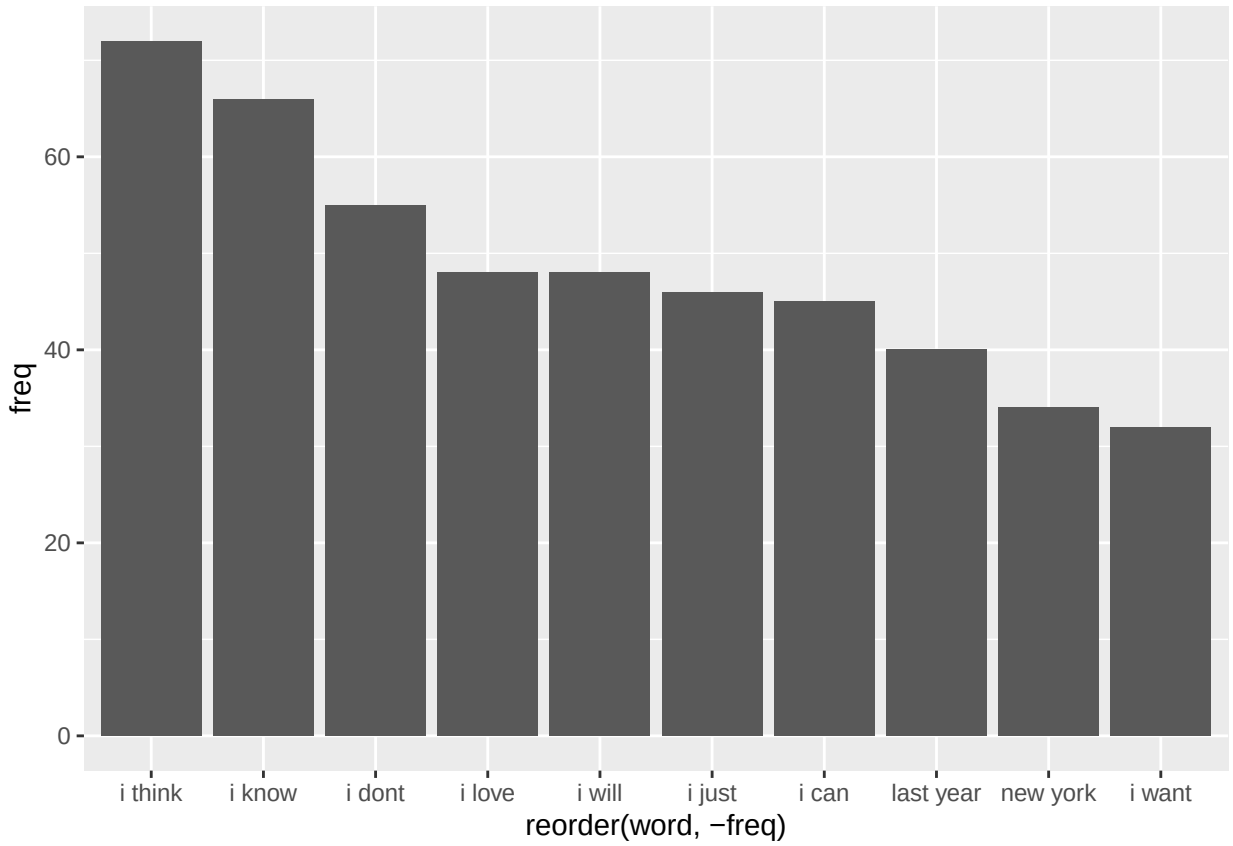
Analysis

tokenizing the data using three ways: 1-gram, 2-gram and 3-gram

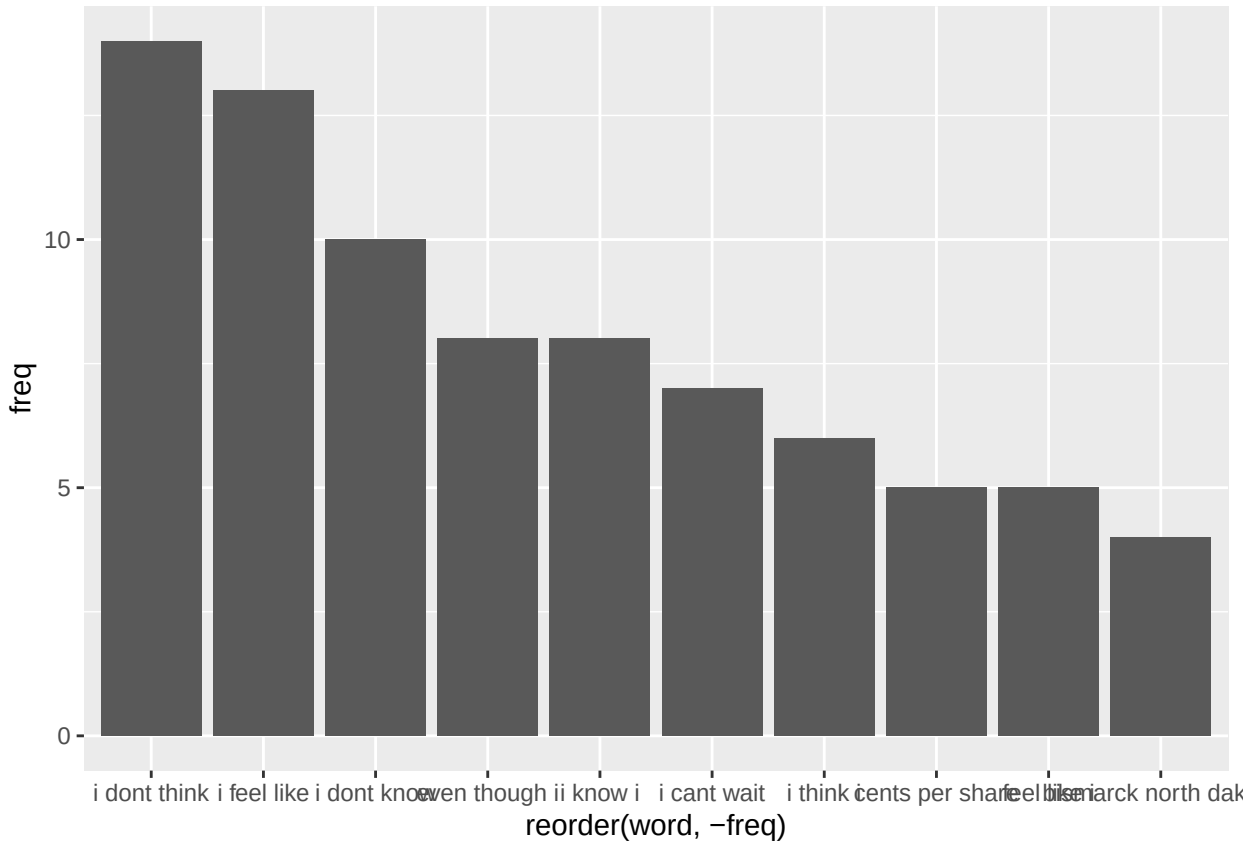
```
# 1 gram
unigram <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
gram1 <- as.data.frame((as.matrix(TermDocumentMatrix(data, control = list(tokenize = unigram)))))
gram1 <- sort(rowSums(gram1),decreasing=TRUE)
gram1 <- data.frame(word = names(gram1),freq=gram1)
data1<- gram1[1:10,]
# visualize the distribution
ggplot(data1, aes(x = reorder(word, -freq), y = freq)) + geom_col()
```




```
# 2 gram
bigram <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
gram2 <- as.data.frame((as.matrix(TermDocumentMatrix(data, control = list(tokenize = bigram)))))
gram2 <- sort(rowSums(gram2),decreasing=TRUE)
gram2 <- data.frame(word = names(gram2),freq=gram2)
data2<- gram2[1:10,]
# visualize the distribution
ggplot(data2, aes(x = reorder(word, -freq), y = freq)) + geom_col()
```




```
# 3 gram
trigram <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
gram3 <- as.data.frame((as.matrix(TermDocumentMatrix(data, control = list(tokenize = trigram)))))
gram3 <- sort(rowSums(gram3),decreasing=TRUE)
gram3 <- data.frame(word = names(gram3),freq=gram3)
data3<- gram3[1:10,]
# visualize the distribution
ggplot(data3, aes(x = reorder(word, -freq), y = freq)) + geom_col()
```



Next step

From the exploratory data analysis, we know the frequencies of each word in blog, news and twitter datasets. We can expand this analysis to more datasets. And after this, we will have a prediction algorithm and an app. The app should contain n-gram algorithem for text prediction. It will take users'input data and give its prediction.