Swift Key :Captone : Data Cleaning and Processing Week1

Kumar Shaket

20/06/2021

Week 1

Loading Natural Language Processing and ggplot packages

##Loading text files after reading file

```
blogfile <-"/Users/kumarshaket/Desktop/Coursera/Capstone Project/final/en_US/en_US.blogs.txt"
newsfile <-"/Users/kumarshaket/Desktop/Coursera/Capstone Project/final/en_US/en_US.news.txt"
twitterfile <- "/Users/kumarshaket/Desktop/Coursera/Capstone Project/final/en_US/en_US.twitter.txt"
bloglines <- readLines(blogfile,encoding = "UTF-8",skipNul = TRUE)
newslines <- readLines(newsfile,encoding = "UTF-8",skipNul = TRUE)
twitterlines <- readLines(twitterfile,encoding = "UTF-8",skipNul = TRUE)
```

2.The en_US.twitter.txt has how many lines of text? Is Over 2 million

```
blogword_count <- length(bloglines)
blogword_count
```

```
## [1] 899288
```

```
twitterword_count <- length(twitterlines)
twitterword_count
```

```
## [1] 2360148
```

```
newsword_count <- length(newslines)
newsword_count
```

```
## [1] 1010242
```

3.What is the length of the longest line seen in any of the three en_US data sets? Is Over 40 thousand in the blogs data set

```
max(nchar(newslines))
```

```
## [1] 11384
```

```
max(nchar(bloglines))
```

```
## [1] 40833
```

4.In the en_US twitter data set, if you divide the number of lines where the word "love" (all lowercase) occurs by the number of lines the word "hate" (all lowercase) occurs, about what do you get?

```
love_count <- sum(grep("love",twitterlines))
hate_count <- sum(grep("hate",twitterlines))
love_count/hate_count
```

```
## [1] 4.111791
```

5.The one tweet in the en_US twitter data set that matches the word "biostats" says what? As below

```
biostat <- grep("biostats",twitterlines)
twitterlines[biostat]
```

```
## [1] "i know how you feel.. i have biostats on tuesday and i have yet to study =/"
```

##6.How many tweets have the exact characters "A computer once beat me at chess, but it was no match for me at kickboxing". (I.e. the line matches those characters exactly.) Is 3

```
sentenceTwitter <- grep("A computer once beat me at chess, but it was no match for me at kickboxing",tw
length(sentenceTwitter)
```

```
## [1] 3
```

Week 2

Sampling the Data

```
set.seed(3000)
stwitter <- sample(twitterlines,size = 2000,replace= TRUE)
sblog <- sample(bloglines,size=2000,replace = TRUE)
snews <- sample(newslines,size = 2000,replace = TRUE)
```

A corpus is created using above sample data from twitter,blog and news

```
corpus <- VCorpus(VectorSource(c(stwitter,snews,sblog)),readerControl = list(readPlain,language="en",loa
```

Exploratory and Cleaning of Data

This section will use the text mining library ‘tm’ (loaded previously) to perform Data cleaning tasks, which are meaningful in Predictive Text Analytics. Main cleaning steps are:

1. Converting the document to lowercase.
2. Removing Whitespace.
3. Removing Punctuation.
4. Removing Numbers 5.Removing stopwords(i.e. “and”,“or”,“not”,“is”)
5. Removing Profanity words from data The above can be achieve with some of the TM package functions; let’s take a look to each cleaning task, individually:
6. Reading profanity words list downloaded from internet and storing in variable

```
badwords <- readLines("/Users/kumarshaket/Desktop/Coursera/Capstone Project/Coursera-Capstone-Project/ba
```

```
## Converting to lower case
corpus_lower <- tm_map(corpus,content_transformer(tolower))
##Removing whitespace from data
corpus_space <- tm_map(corpus_lower,stripWhitespace)
## Removing Punctuation from data
corpus_punc <- tm_map(corpus_space,removePunctuation)
## Removing Number from data
corpus_numb<- tm_map(corpus_punc,removeNumbers)
## Removint stop words from data
corpus_nostop <- tm_map(corpus_numb,removeWords,stopwords("english"))
## Removing profanity words from data
finalCorpus <- tm_map(corpus_nostop,removeWords,badwords)
```

Tokenization

Let’s read the text to break it into words and sentences, and to turn it into n-grams. These are all called tokenization because we are breaking up the text into units of meaning, called tokens.

In Natural Language Processing (NLP), n-gram is a contiguous sequence of n items from a given sequence of text or speech. Unigrams are single words. Bigrams are two words combinations. Trigrams are three-word combinations.

The tokenizer method is allowed in R using the package RWeka. The following function is used to extract unigram,bigram,trigram from the text Corpus using RWeka.

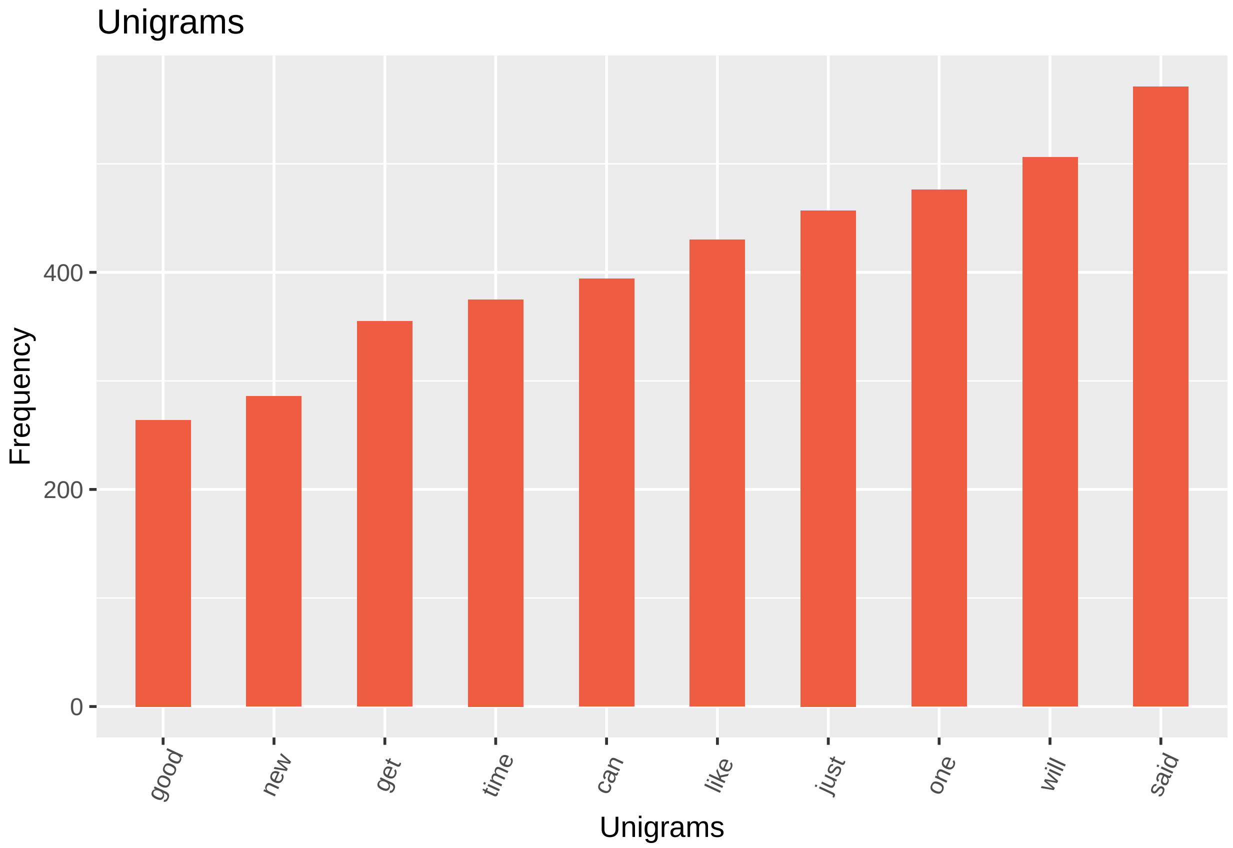
Obtaining Unigram

```
unigram <- as.data.frame((as.matrix(TermDocumentMatrix(finalCorpus))))
unigram1v <- sort(rowSums(unigram),decreasing = TRUE)
unigram1d <- data.frame(word=names(unigram1v),freq=unigram1v)
unigram1d[1:10,]
```

```
##      word freq
## said said  571
## will will  506
## one   one  476
## just just  457
## like like  430
## can   can  394
## time time  375
## get   get  355
## new   new  286
## good good  264
```

Histogram of top 10 Unigrams

```
ggplot(unigram1d[1:10,],aes(x=reorder(word,freq),y=freq))+
        geom_bar(stat = "identity",width = 0.5,fill="tomato2") +
        labs(title = "Unigrams")+
        xlab("Unigrams")+ylab("Frequency")+
        theme(axis.text.x = element_text(angle = 65,vjust = 0.6))
```



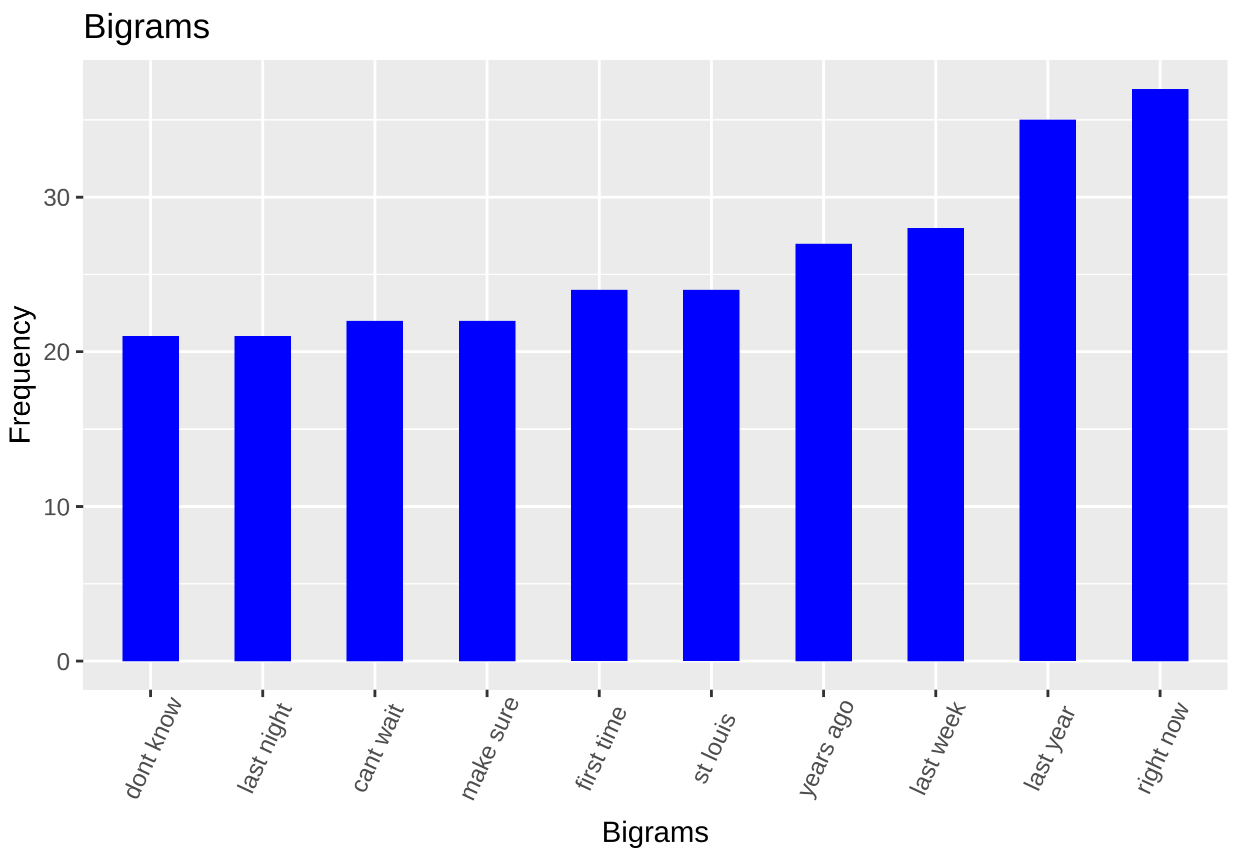
Obtaining BiGram

```
bigram <- function(x) NGramTokenizer(x,Weka_control(min=2,max=2))
gram2 <- as.data.frame(as.matrix(TermDocumentMatrix(finalCorpus,control = list(tokenize=bigram))))
gram2v <- sort(rowSums(gram2),decreasing = TRUE)
gram2d <- data.frame(word=names(gram2v),freq=gram2v)
gram2d[1:10,]
```

```
##                  word freq
## right now   right now   37
## last year   last year   35
## last week   last week   28
## years ago   years ago   27
## first time first time   24
## st louis      st louis   24
## cant wait   cant wait   22
## make sure   make sure   22
## dont know   dont know   21
## last night last night   21
```

Histogram of top 10 BiGram

```
ggplot(gram2d[1:10,],aes(x=reorder(word,freq),y=freq))+
        geom_bar(stat = "identity",width = 0.5,fill="blue") +
        labs(title = "Bigrams")+
        xlab("Bigrams")+ylab("Frequency")+
        theme(axis.text.x = element_text(angle = 65,vjust = 0.6))
```



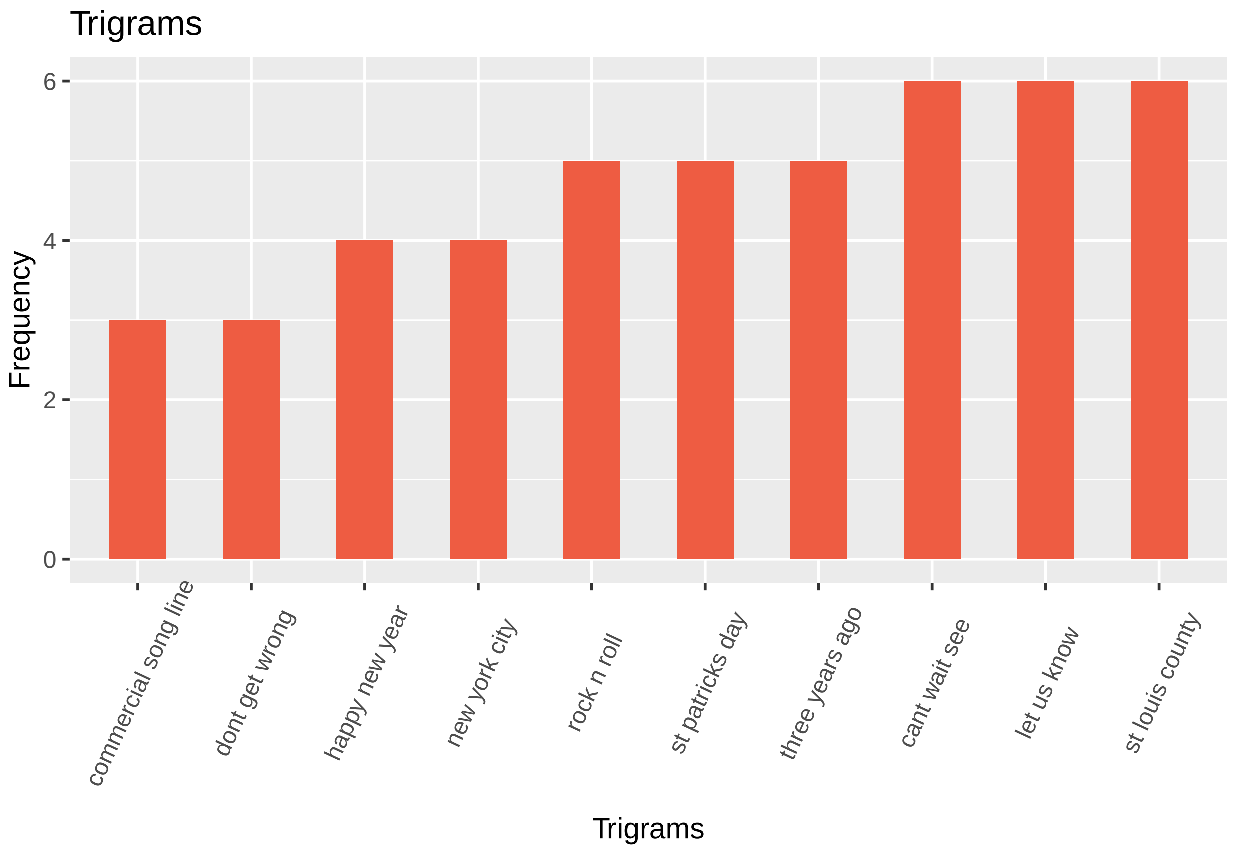
### Obtaining Trigram

```
trigram <- function(x) NGramTokenizer(x,Weka_control(min=3,max=3))
gram3 <- as.data.frame(as.matrix(TermDocumentMatrix(finalCorpus,control = list(tokenize=trigram))))
gram3v <- sort(rowSums(gram3),decreasing = TRUE)
gram3d <- data.frame(word=names(gram3v),freq=gram3v)
gram3d[1:10,]
```

```
##                                        word freq
## cant wait see                 cant wait see    6
## let us know                     let us know    6
## st louis county             st louis county    6
## rock n roll                     rock n roll    5
## st patricks day             st patricks day    5
## three years ago             three years ago    5
## happy new year               happy new year    4
## new york city                 new york city    4
## commercial song line commercial song line    3
## dont get wrong               dont get wrong    3
```

Histogram Of 10 Top Trigrams

```
ggplot(gram3d[1:10,],aes(x=reorder(word,freq),y=freq))+
        geom_bar(stat = "identity",width = 0.5,fill="tomato2") +
        labs(title = "Trigrams")+
        xlab("Trigrams")+ylab("Frequency")+
        theme(axis.text.x = element_text(angle = 65,vjust = 0.6))
```



Next Steps

1. Build a Shiny app to allow the user input the word to obtain a suggestion of the next word.
2. Develop the prediction algorithm implemented in Shiny app.
3. Prepare a pitch about the app and publish it at “shinyapps.io” server.