Data Science Capstone

Himank Jain

24/09/2019

N Gram model for word prediction:

Introduction & Understanding the Problem:

Around the world, people are spending an increasing amount of time on their mobile devices for email, social networking, banking and a whole range of other activities. But typing on mobile devices can be a serious pain. SwiftKey, our corporate partner in this capstone, builds a smart keyboard that makes it easier for people to type on their mobile devices. One cornerstone of their smart keyboard is predictive text models. When someone types:

I went to the

the keyboard presents three options for what the next word might be. For example, the three words might be gym, store, restaurant. Here, We build Predictive Ngram (2-gram, 3-gram, 4-gram, and 5-gram) models based on Katz's Back off model and integrate it in an application which is the end product.

This project is the capstone project of Data Science Specialization course provided by JHU on Coursera.

The final Application predicts next word, given a set of words by a user as input. It's hosted on shinyapps.io Click here to directly go to the Application.

Reproducability:

```
date() #project start date: 25th september,2019
```

```
## [1] "Sat Oct 12 13:08:15 2019"
```

```
set.seed(369)
```

This Document contains:

- Exploratory analysis - performing a thorough exploratory analysis of the data, understanding the distribution of words and relationship between the words in the corpora.
- Understanding frequencies of words and word pairs - build figures and tables to understand variation in the frequencies of words and word pairs in the data.
- Ngram models
- Katz's Back off model

Exploratory Analysis:

The first step in building a predictive model for text is understanding the distribution and relationship between the words, tokens, and phrases in the text. The goal of this task is to understand the basic relationships we observe in the data and prepare to build our linguistic models.

setting up environment:

```
# Required R libraries:
library(stringi)
library(ggplot2)
library(data.table)
library(htmlwidgets)
library(magrittr)
library(webshot)
#webshot::install_phantomjs()
library(markdown)
library(RWeka)
library(openNLP)
library(wordcloud2)
library(wordcloud)
library(tm)
library(NLP)
library(qdap)
library(devtools)
library(plotrix)
```

Data Acquisition and cleaning:

Dataset

The data is provided by JHUCapstone Course in Data Science. It can be downloaded from here

NOTE: Since the data files are relatively big, they are not present in this repository. Download them from the link above.

Reading files & Sampling:

```
#reading few lines from files
news<-readLines("final/en_US/en_US.news.txt")
blogs<-readLines("final/en_US/en_US.blogs.txt")
twitter<-readLines("final/en_US/en_US.twitter.txt")


#sampling only 5000 lines from each file for performance efficiency:
fewnews<-sample(news,5000)
fewblogs<-sample(blogs,5000)
fewtwit<-sample(twitter,5000)

#combining text from the three files
fewtext<-paste(fewnews,fewblogs,fewtwit)
fewtext[[1]]
```

```
## [1] "Gov. Tom Corbett, also a Republican, says he'll sign it. It received Senate approval last week.
```

Summarization:

```r
sizeblogs<-file.size('final/en_US/en_US.blogs.txt')/1024^2
sizenews<-file.size('final/en_US/en_US.news.txt')/1024^2
sizetwitter<-file.size('final/en_US/en_US.twitter.txt')/1024^2

wordsblogs<-sum(sapply(strsplit(blogs, " "), length))
wordsnews<-sum(sapply(strsplit(news, " "), length))
wordstwitter<-sum(sapply(strsplit(twitter, " "), length))


charsblogs<-sum(nchar("blogs"))
charsnews<-sum(nchar("news"))
charstwitter<-sum(nchar("twitter"))

linesblogs<-length("blogs")
linesnews<-length("news")
linestwitter<-length("twitter")
# Displaying details of files:
data.frame(Name=c("US_blogs", "US_news", "US_twitter"),
           Lines=c(linesblogs,linesnews,linestwitter),
           Words=c(wordsblogs,wordsnews,wordstwitter),
           Size_in_MB=c(sizeblogs,sizenews,sizetwitter))
```

```
##         Name Lines    Words Size_in_MB
## 1   US_blogs     1 37334131   200.4242
## 2    US_news     1  2643969   196.2767
## 3 US_twitter     1 30373590   159.3641
```

Preprocessing:

```r
#making corpus
corpus<-VCorpus(VectorSource(fewtext))
# remove numbers
corpus <- tm_map(corpus, removeNumbers)
 # strip whitespaces left and right
corpus <- tm_map(corpus, stripWhitespace)
# convert all chars to lowercase
corpus <- tm_map(corpus, content_transformer(tolower))
# remove special characters
corpus <- tm_map(corpus, removePunctuation)

#getting rid profane words:
badwords<-file('badwords.txt')
badwords<-VectorSource(badwords)
corpus <- tm_map(corpus, removeWords, badwords)
corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5000
```

Note: Consider removing stopwords as they do not provide any descriptive information. Stopwords haven't been removed here.

```
#corpus<-tm_map(corpus,removeWords,stopwords(language='english'))
```

Stemming with english language(only for exploratory analysis):

```
corpus<-tm_map(corpus,stemDocument,language='english')
cleanData<-data.table(text=(sapply(corpus,'[','content')),stringsAsFactors = F)
```

Exploratory Analysis:

ngram Tokenizaton for 1,2,3 and 4 grams:

```
unitok<-NGramTokenizer(cleanData,Weka_control(min=1,max=1))

bitok <- NGramTokenizer(cleanData, Weka_control(min = 2, max = 2, delimiters = " \\r\\n\\t.
,;:\"()?!"))
tritok <- NGramTokenizer(cleanData, Weka_control(min = 3, max = 3, delimiters = " \\r\\n\\t
.,;:\"()?!"))
head(unitok)
```

```
## [1] "list"    "gov"     "tom"     "corbett" "also"    "a"
```

```
tetratok<-NGramTokenizer(cleanData,Weka_control(min=4,max=4,delimiters = " \\r\\n\\t
.,;:\"()?!"))
head(bitok)
```

```
## [1] "list gov"     "gov tom"      "tom corbett"  "corbett also"
## [5] "also a"       "a republican"
```

```
head(tritok)
```

```
## [1] "list gov tom"      "gov tom corbett"   "tom corbett also"
## [4] "corbett also a"    "also a republican" "a republican say"
```

Q:What are the frequencies of 2-grams and 3-grams in the dataset?

```
unidf<-data.table(table(unitok))
unidf<-unidf[order(-N),]
colnames(unidf)<-c('word','freq')

bitokdf<-data.table(table(bitok))
```

```
bitokdf<-bitokdf[order(-N),]
colnames(bitokdf)<-c('word','freq')

tritokdf<-data.table(table(tritok))
tritokdf<-tritokdf[order(-N),]
colnames(tritokdf)<-c('word','freq')

tetratokdf<-data.table(table(tetratok))
tetratokdf<-tetratokdf[order(-N),]
colnames(tetratokdf)<-c('word','freq')
bitokdf[1:15,]
```

```
##          word freq
##  1:    of the 2062
##  2:    in the 1915
##  3:    to the  972
##  4:   for the  876
##  5:    on the  875
##  6:     to be  706
##  7:   and the  623
##  8:    at the  598
##  9:      in a  563
## 10:  with the  485
## 11:    it was  452
## 12:   want to  433
## 13:     for a  431
## 14:      is a  427
## 15:  from the  420
```

```
tritokdf[1:15,]
```

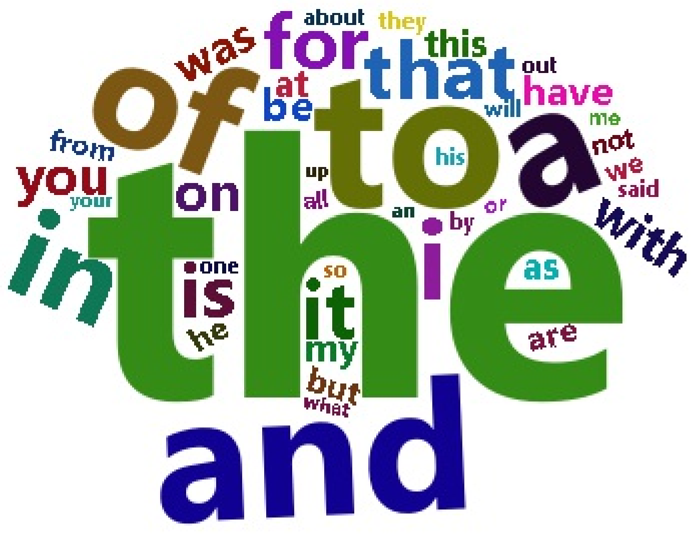
```
##                word freq
##  1:      one of the  168
##  2:        a lot of  130
##  3:       i want to  109
##  4:      the end of   80
##  5:     part of the   76
##  6:        it was a   75
##  7:       be abl to   73
##  8:         to be a   72
##  9:     the rest of   70
## 10:      out of the   66
## 11:        go to be   65
## 12:     some of the   62
## 13:      a coupl of   61
## 14:      as well as   56
## 15: look forward to   56
```

Q: Some words are more frequent than others - what are the distributions of word frequencies?:

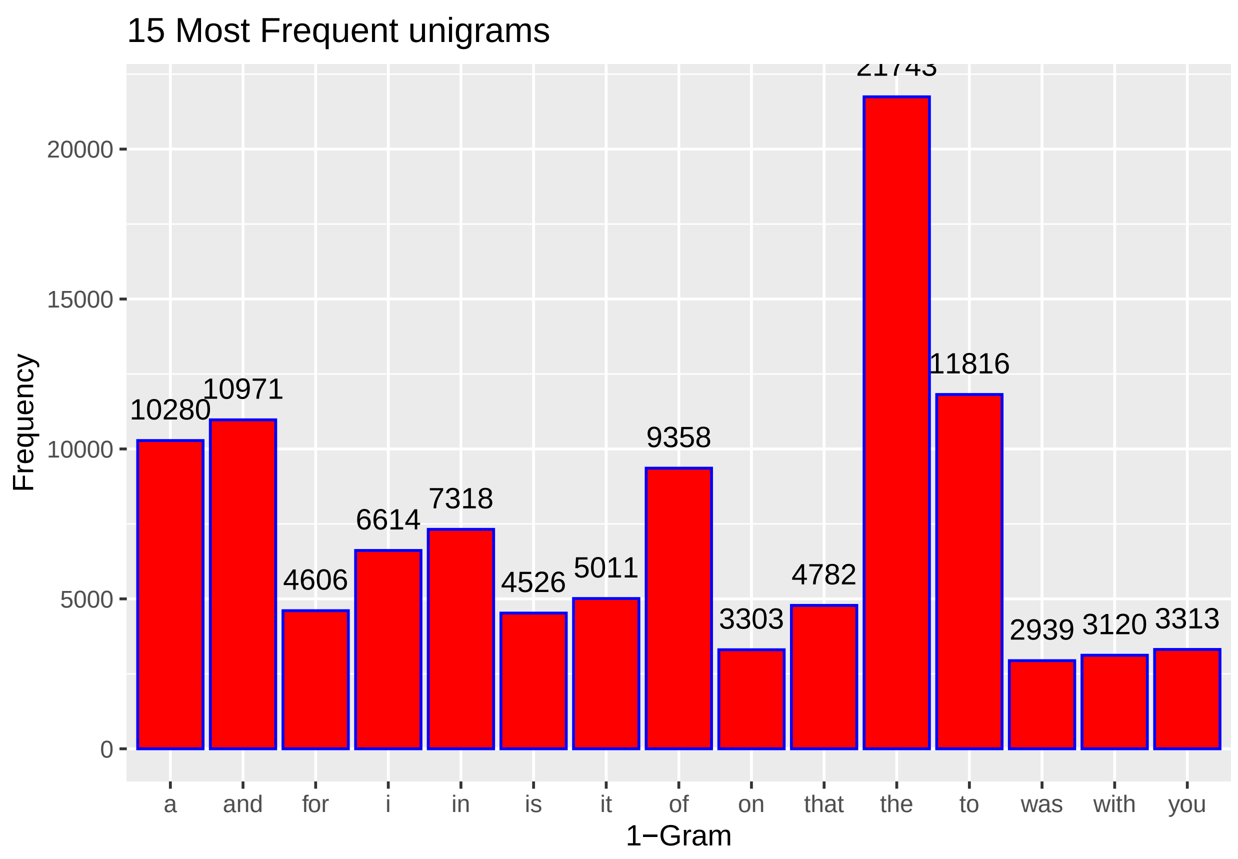
WORD CLOUDS & Distributions:

1-Gram

```
wordcloud2(unidf[1:2000,])
```

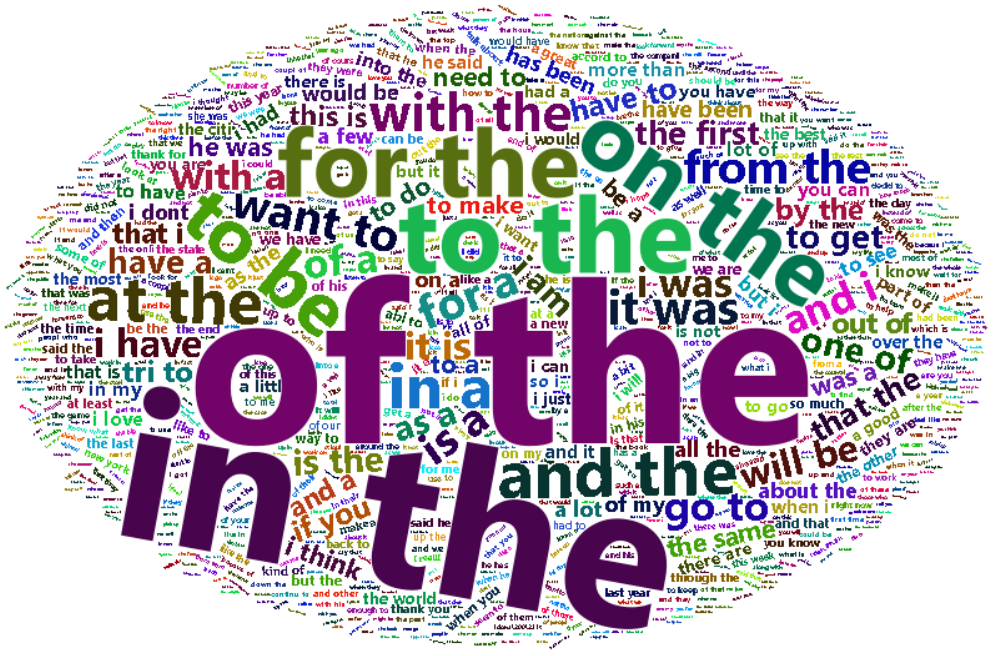


```
splot<-ggplot(unidf[1:15],aes(x=word,y=freq))+geom_bar(stat='identity',fill='red',colour='blue')+
  geom_text(aes(label=freq),vjust=-1)+labs(title='15 Most Frequent unigrams',x='1-Gram',y='Frequency')
splot
```



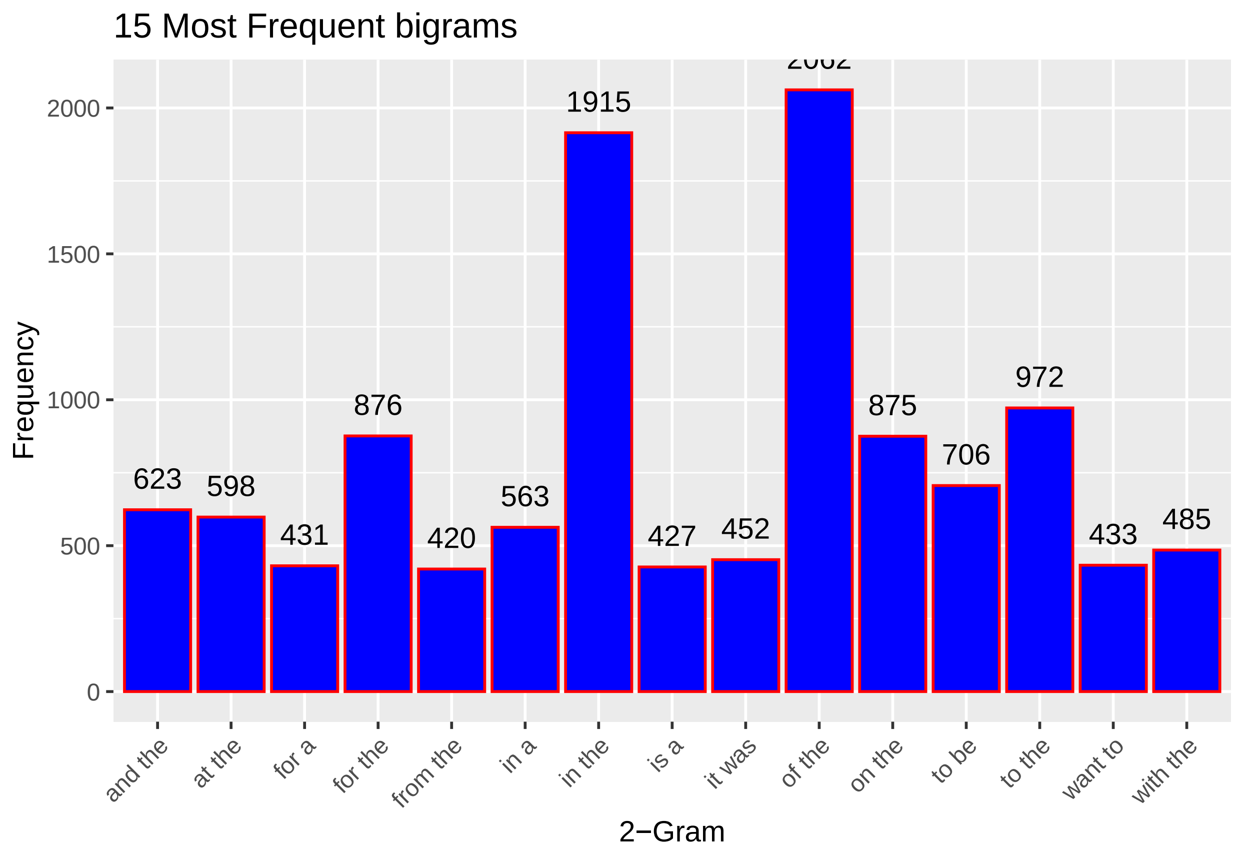
### 2-Gram

```
wc2<-wordcloud2(bitokdf[1:1000,])
saveWidget(wc2,"2.html",selfcontained = F)
webshot::webshot("2.html","2.png",vwidth = 1000, vheight = 800, delay =10)
```



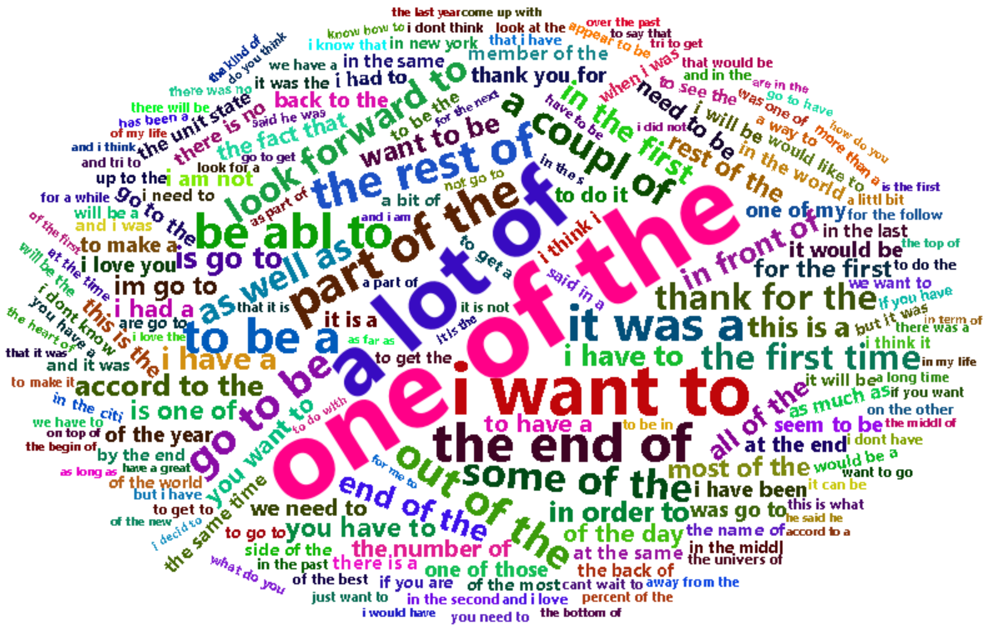
```
bplot<-ggplot(bitokdf[1:15],aes(x=word,y=freq))+geom_bar(stat='identity',fill='blue',colour='red')+geom_
  theme(axis.text.x=element_text(angle = 45,hjust=1))+labs(title='15 Most Frequent bigrams',x='2-Gram',y
```

```
bplot
```



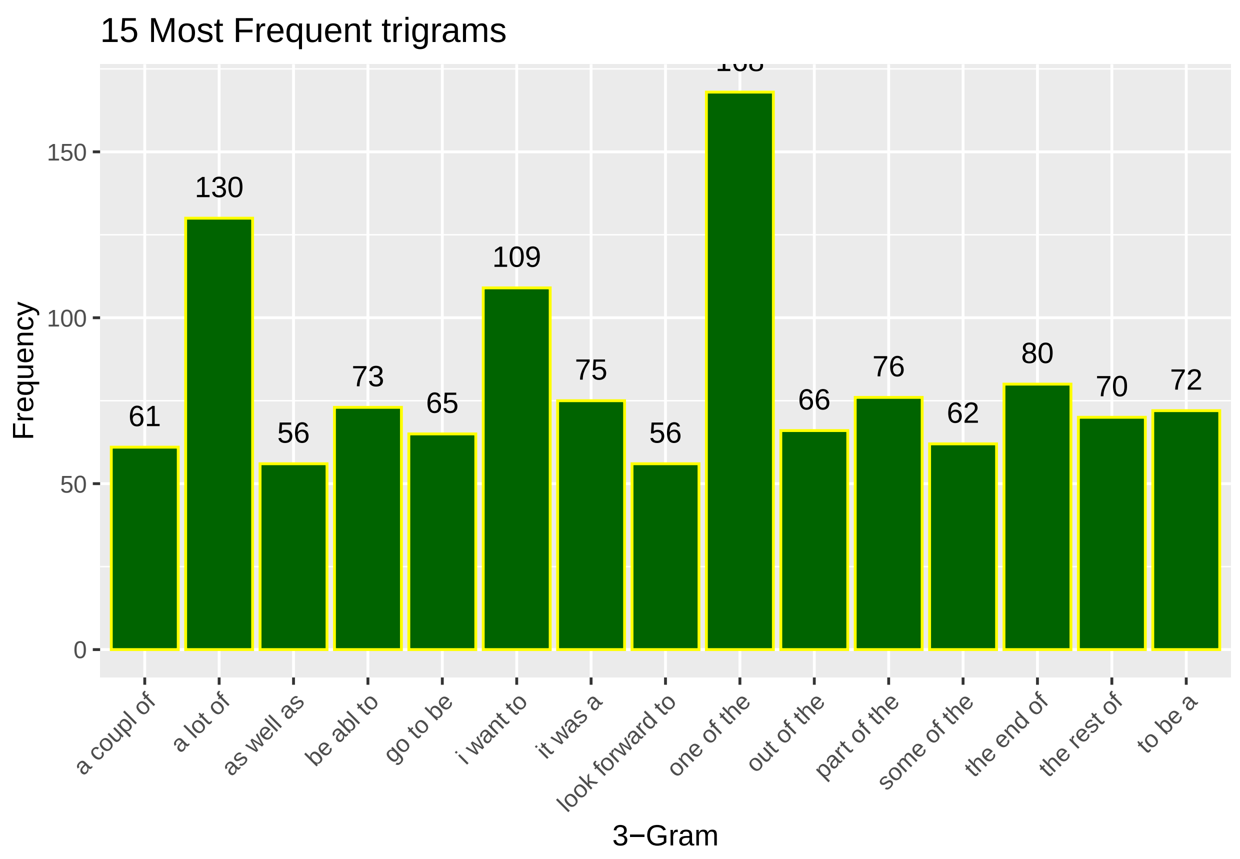
3-Gram

```
wc3<-wordcloud2(tritokdf[1:200],minSize = 0.001,size=0.5)
saveWidget(wc3,"3.html",selfcontained = F)
webshot::webshot("3.html","3.png",vwidth = 1000, vheight = 800, delay =10)
```



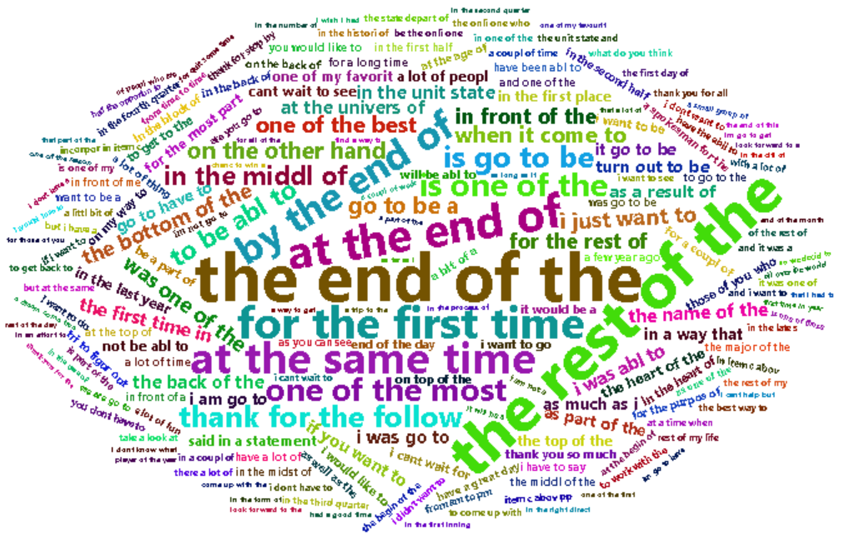
```
tplot<-ggplot(tritokdf[1:15],aes(x=word,y=freq))+geom_bar(stat='identity',fill='dark green',colour='yel
  theme(axis.text.x=element_text(angle = 45,hjust=1))+labs(title='15 Most Frequent trigrams',x='3-Gram'
```

```
tplot
```



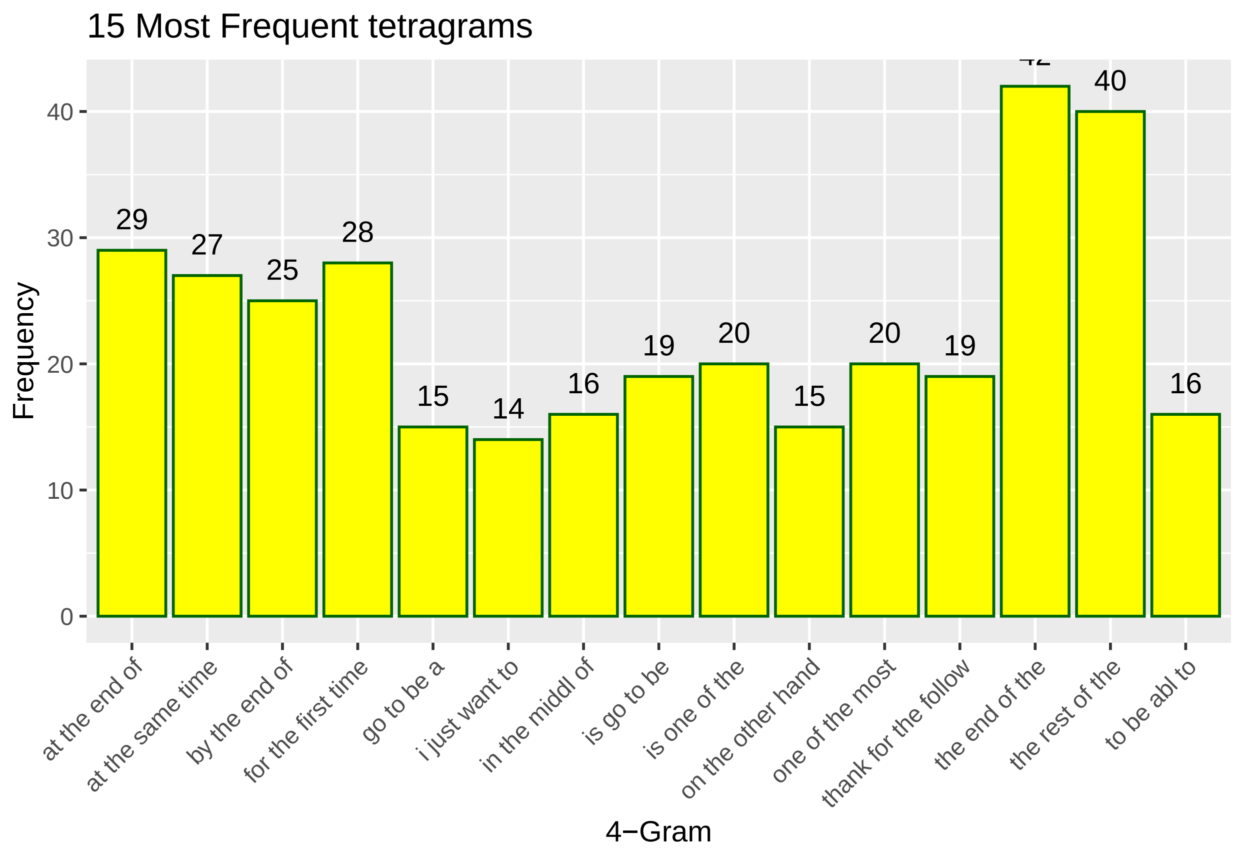
4-Gram

```
wc4<-wordcloud2(tetratokdf[1:200],minSize=0.001,size=0.3)
saveWidget(wc4,"4.html",selfcontained = F)
webshot::webshot("4.html","4.png",vwidth = 1000, vheight = 800, delay =10)
```



```
ttplot<-ggplot(tetratokdf[1:15],aes(x=word,y=freq))+geom_bar(stat='identity',fill='yellow',colour='dark
  theme(axis.text.x=element_text(angle = 45,hjust=1))+labs(title='15 Most Frequent tetragrams',x='4-Gra
```

```
ttplot
```



How many unique words do we need in a frequency sorted dictionary to cover 50% of all word instances in the language? 90%?

Creating a function to get minimum numbers of tokens required to explain given frequence percentage

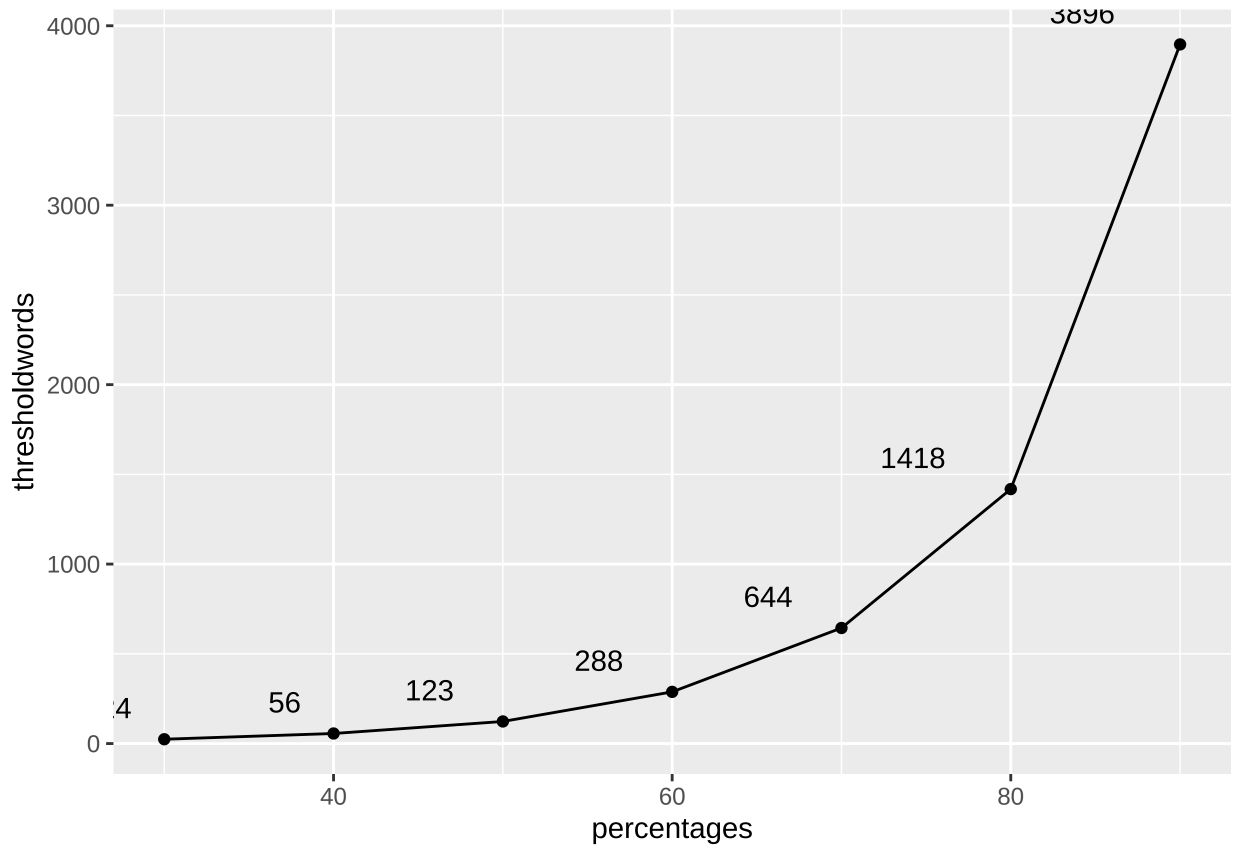
```
explained<-function(df,percentage){
  cumperc=0
  n=0
  totalfreq=sum(df$freq)
  for (i in 1:nrow(df)){
    perc=(df$freq[i]/totalfreq)*100
    n=n+1
    cumperc=cumperc+perc
    if(cumperc>=percentage){
      return(n)
    }
  }
}

explained(unidf,50)
```

```
## [1] 123
```

Hence, we need only Top 123 tokens to cover 50% of the total frequency!

Frequency covered by using minimum number of tokens:

```
percentages<-seq(30,90,10)
thresholdwords<-c(explained(unidf,30),explained(unidf,40),explained(unidf,50),explained(unidf,60),expla
qplot(percentages,thresholdwords,geom=c('line','point'))+geom_text(aes(label=thresholdwords),hjust=2,vj
```



Around 3900 tokens for covering 90% of total frequency.

Q:How do we evaluate how many of the words come from foreign languages?

We can evaluate this using a dictionary to remove words that are not from english language or that are just typos. And then calculate the number of words removed. We can also use different languages dictionaries to compare removed words but that would be computationally expensive.

Q:Can we think of a way to increase the coverage – identifying words that may not be in the corpora or using a smaller number of words in the dictionary to cover the same number of phrases?

We can increase the coverage by converting all words to lower letter. Also we can use stemming which reduces words into their respective word stems which significantly increases frequency of common words and also reduces number of words in dictionary.

Modelling Plan & Shiny Application:

The goal here is to build the first simple model for the relationship between words. This is the first step in building a predictive text mining application. We will explore simple models and discover more complicated modeling techniques.

Tasks to accomplish

- Build basic n-gram model - using the exploratory analysis we performed, build a basic n-gram model for predicting the next word based on the previous 1, 2, or 3 words.
- Build a model to handle unseen n-grams - in some cases people will want to type a combination of words that does not appear in the corpora. Building a model to handle cases where a particular n-gram isn't observed.

We have already Built 1-gram,2-gram, 3-gram and 4-gram dictionaries for our model.

- We would have to:
  - Efficiently store the n-gram model
  - Reduce the *size* and *runtime* of the model
  - use appropriate n in our ngram model. Here we are using 1,2,3 and 4
  - Smooth out the probabilities
  - Evaluate efficiency and accuracy of our model
  - Use backoff models to deal with unobserved ngrams
  - Also,We would build our model taking into account that currently available predictive text models can run on mobile phones, which typically have limited memory and processing power compared to desktop computers.
  - At last, we would create a ShinyApp that runs our model by taking input from the user and predicting next word based on our ngram model.
  - The shinyApp would run on shinyapps.io server.

Predictive Model :

Tasks Accomplished:

- Build a predictive model based on the previous data modeling steps - you may combine the models in any way you think is appropriate.
- Evaluate the model for efficiency and accuracy - use timing software to evaluate the computational complexity of your model. Evaluate the model accuracy using different metrics like perplexity, accuracy at the first word, second word, and third word.

NOTE: i used tm package for exploratory analysis. For modelling However, i used quanteda package since it's computationally more efficient

Tokenization and Cleaning has been done in tokenizationandcleaning.R file.

The Ngrams have been computed in ngrams.R file

A function called ngrams is created in prediction.R file which predicts next word given an input string. It uses output from ngram.R file

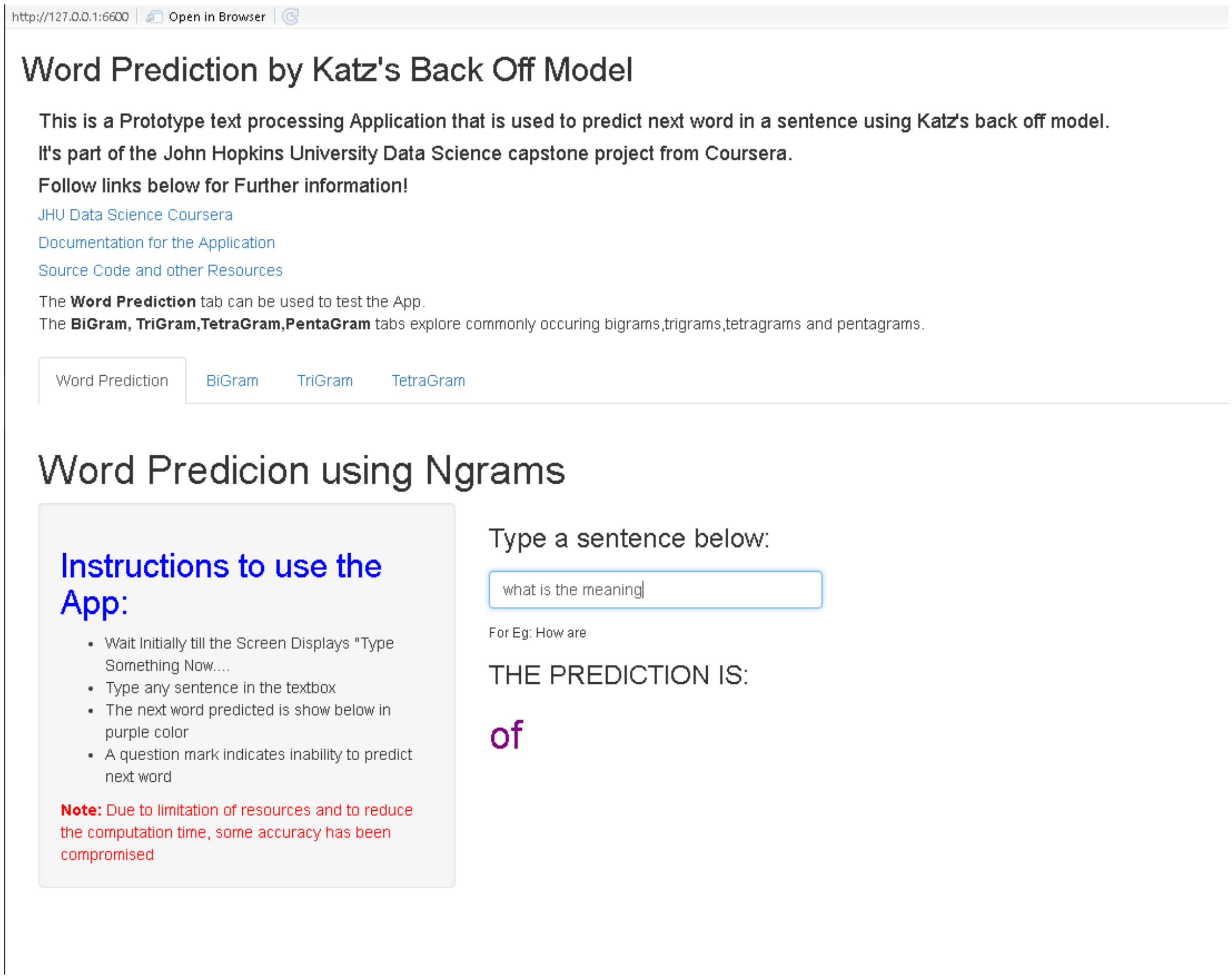
Final Application, Documentation & Resources:

Finally, After model building I used R shinyApp interface to integrate the katz's back off model to build a predictive application that is hosted on shinyapps.io

Applicaton:

Final Application

Application Image Demo:



http://127.0.0.1:6600 Open in Browser

Word Prediction by Katz's Back Off Model

This is a Prototype text processing Application that is used to predict next word in a sentence using Katz's back off model.

It's part of the John Hopkins University Data Science capstone project from Coursera.

Follow links below for Further information!

JHU Data Science Coursera

Documentation for the Application

Source Code and other Resources

The **Word Prediction** tab can be used to test the App.
The **BiGram, TriGram,TetraGram,PentaGram** tabs explore commonly occuring bigrams,trigrams,tetragrams and pentagrams.

Word Predicion using Ngrams

Instructions to use the App:

- Wait Initially till the Screen Displays "Type Something Now....
- Type any sentence in the textbox
- The next word predicted is show below in purple color
- A question mark indicates inability to predict next word

Note: Due to limitation of resources and to reduce the computation time, some accuracy has been compromised

Type a sentence below:

what is the meaning

For Eg: How are

THE PREDICTION IS:

of

Pitch Deck:

Pitch

Resources and files:

Project Github Repo

Thank You