Data Science Capstone

Jamin Wong

2022-07-21

Abstract

In this project, I will use the data set provided by Coursera and Swift Key make a Shiny website. It will divided several parts: Understanding the problem

-Data acquisition and cleaning

-Exploratory analysis

-Statistical modeling

-Predictive modeling

-Creative exploration

-Creating a data product

-Creating a short slide deck pitching your product

The files in downloaded from: https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip I will be using the files in final/en_US folder mainly

Getting Data

Load file

```
fileB <- readLines("final/en_US/en_US.blogs.txt")
fileN <- readLines("final/en_US/en_US.news.txt")
```

```
## Warning in readLines("final/en_US/en_US.news.txt"): incomplete final line found
## on 'final/en_US/en_US.news.txt'
```

```
fileT <- readLines("final/en_US/en_US.twitter.txt")
```

```
## Warning in readLines("final/en_US/en_US.twitter.txt"): line 167155 appears to
## contain an embedded nul

## Warning in readLines("final/en_US/en_US.twitter.txt"): line 268547 appears to
## contain an embedded nul

## Warning in readLines("final/en_US/en_US.twitter.txt"): line 1274086 appears to
## contain an embedded nul

## Warning in readLines("final/en_US/en_US.twitter.txt"): line 1759032 appears to
## contain an embedded nul
```

```
fileA <- rbind(fileB, fileN, fileT)
```

```
## Warning in rbind(fileB, fileN, fileT): number of columns of result is not a
## multiple of vector length (arg 1)
```

Summary

```
summ <- sapply(list(fileB, fileN, fileT), stri_stats_general)
wdctA <- sapply(list(fileB, fileN, fileT), wordcount)
rbind(c("blogs", "news", "twitter"), summ, wdctA)
```

```
##             [,1]        [,2]       [,3]
##             "blogs"     "news"     "twitter"
## Lines       "899288"    "77259"    "2360148"
## LinesNEmpty "899288"    "77259"    "2360148"
## Chars       "206824382" "15639408" "162096031"
## CharsNWhite "170389539" "13072698" "134082634"
## wdctA       "37334131"  "2643969"  "30373543"
```

Sample

Create sample with only a small portion of the original to reduce the processing time while keeping the accuracy of the result model. ### Create Sample

```
p <- 0.05
```

It will use 0.05 of the original data set.

```
samp <- sample(fileA, size = round(length(fileA) * p))
```

The sample is taken from random sampling with size 0.05‘ of the original. This is to reduce the file size and processing.

Save Sample

```
writeLines(samp, "sample.txt")
```

Create a sample once and then load it from the sample.txt after the first time

```
## Warning in rm(list = c("fileB", "fileA", "fileT", "fileT", "samp")): object
## 'fileT' not found
```

```
## Warning in rm(list = c("fileB", "fileA", "fileT", "fileT", "samp")): object
## 'samp' not found
```

Load Sample

```
sample_txt <- readLines("sample.txt")
```

The sample is taken from random sampling with size 0.05 of the original

Sample Summary

```
wdct <- wordcount(sample_txt)
cbind(t(stri_stats_general(sample_txt)), fileSize = format(object.size(sample_txt),
    "Mb"), wordCount = wdct)
```

```
##      Lines    LinesNEmpty Chars      CharsNWhite fileSize  wordCount
## [1,] "354022" "354022"    "58988913" "48914807"  "64.2 Mb" "10428127"
```

Cleaning data

Remove URL

```
sample_txt <- gsub("http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+",
    "", sample_txt)
```

Remove punctuation

```
sample_txt <- sample_txt %>%
    removePunctuation()
```

Remove numbers

```
sample_txt <- sample_txt %>%
    removeNumbers()
```

Change all to lower case

```
sample_txt <- sample_txt %>%
    tolower()
```

Remove extra white space

```
sample_txt <- sample_txt %>%
    stripWhitespace()
```

Exploratory Analysis

In the exploratory analysis, I will be using n grams to find out the common phrases

N-grams

Unigram

```
sample_txt <- data.frame(text = sample_txt)
unigram <- sample_txt %>%
    unnest_tokens(word, text)
```

Unigram phrase table with proportion

```
uniPt <- unigram %>%
    count(word, sort = TRUE) %>%
    mutate(prop = n/sum(n))
```

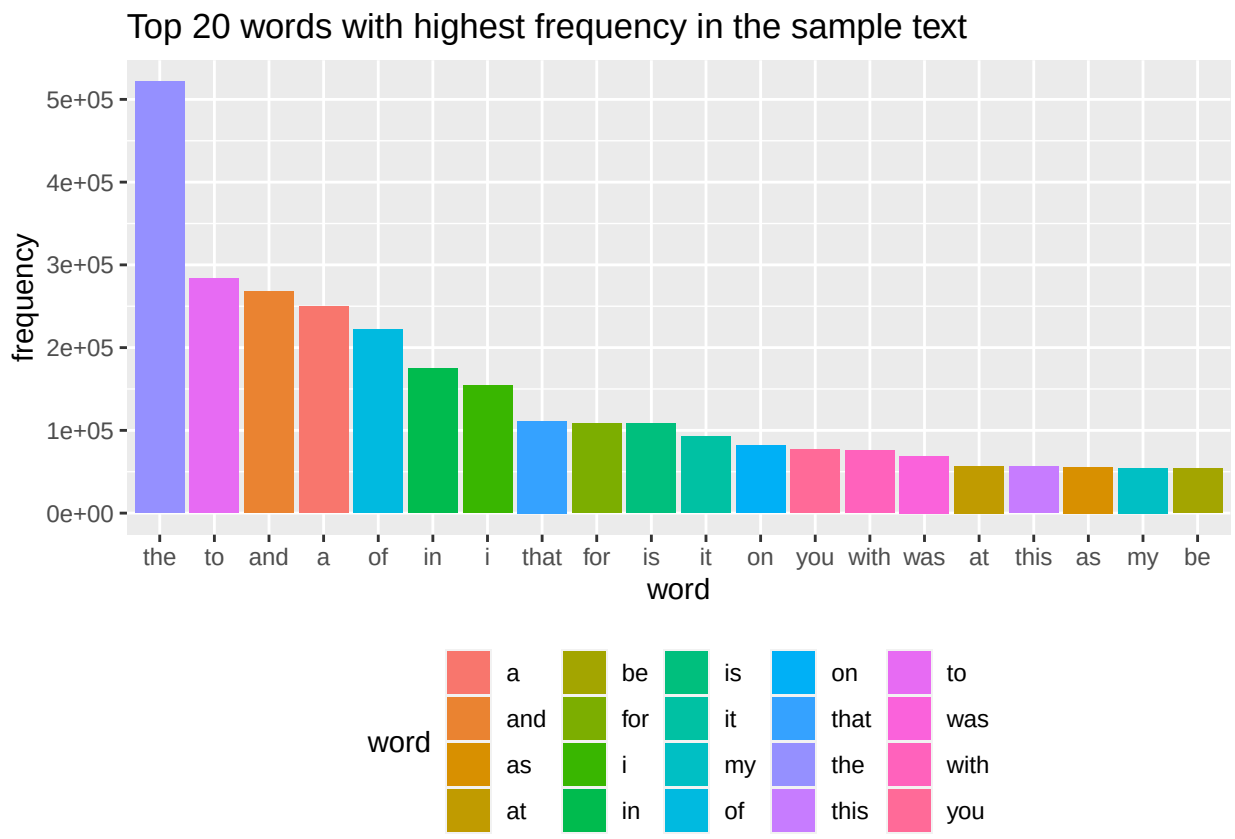
```
head(uniPt, 20)
```

```
##    word      n        prop
## 1   the 521218 0.051067236
## 2    to 283713 0.027797272
## 3   and 267926 0.026250514
## 4     a 249546 0.024449701
## 5    of 221941 0.021745054
## 6    in 174653 0.017111930
## 7     i 154042 0.015092532
## 8  that 111311 0.010905888
## 9   for 108463 0.010626850
## 10   is 108336 0.010614407
## 11   it  93010 0.009112816
## 12   on  81606 0.007995489
## 13  you  76830 0.007527552
## 14 with  75997 0.007445938
## 15  was  68894 0.006750009
## 16   at  56951 0.005579873
## 17 this  56434 0.005529219
## 18   as  55633 0.005450740
## 19   my  54628 0.005352273
## 20   be  54332 0.005323272
```

top 20 words in unigram

```
g1 <- ggplot(uniPt[1:20, ], aes(x = reorder(word, -n), y = n,
    fill = word))
g1 <- g1 + geom_bar(stat = "identity") + labs(x = "word", y = "frequency",
```

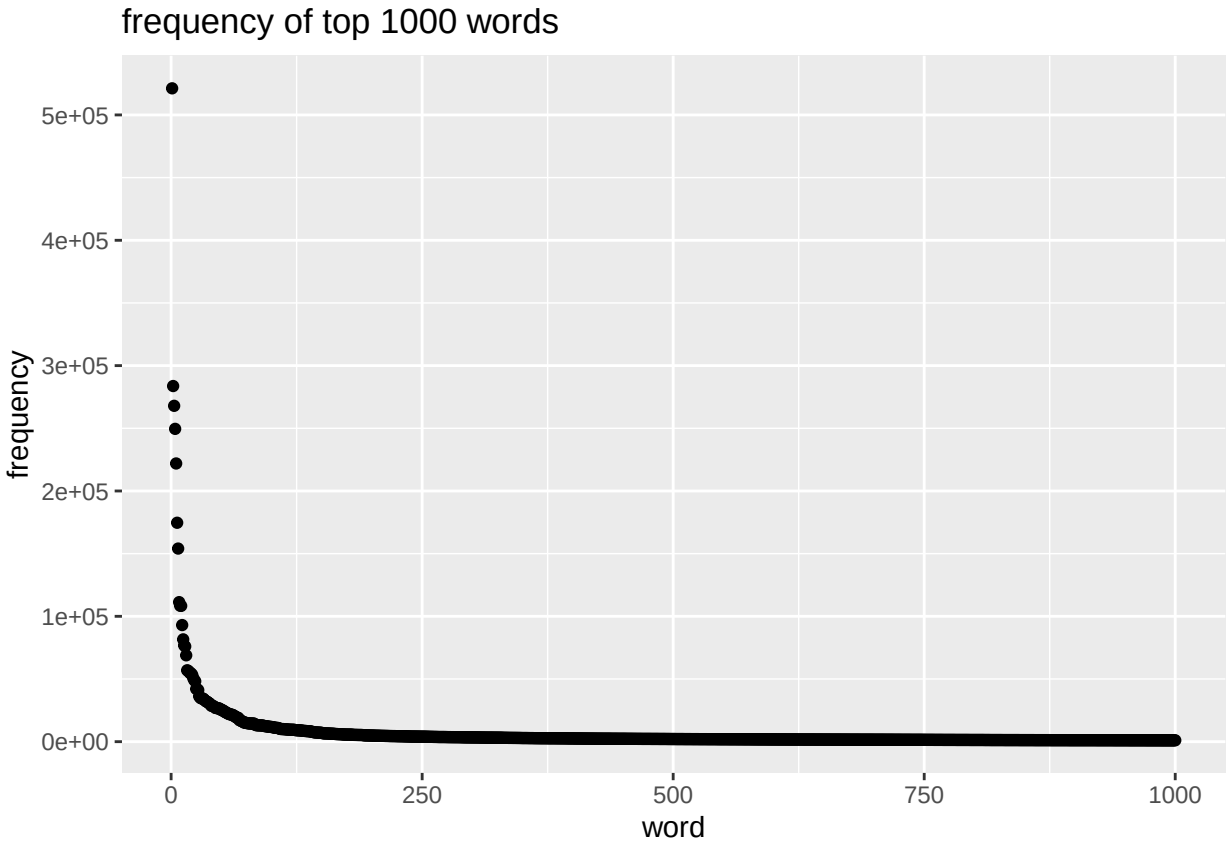
```
    title = "Top 20 words with highest frequency in the sample text") +
    theme(legend.position = "bottom")
g1
```



"the", "to", and "and" have the three highest frequency in the unigrams.

frequency of top 1000 words

```
g4 <- ggplot(uniPt[1:1000, ], aes(x = as.numeric(row.names(uniPt[1:1000,
    ])), y = n))
g4 <- g4 + geom_point() + labs(x = "word", y = "frequency", title = "frequency of top 1000 words")
g4
```


frequency of top 1000 words
frequency
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
0
250
500
750
1000
word

Bigram

```
bigram <- sample_txt %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)
```

Bigram phrase table with proportion

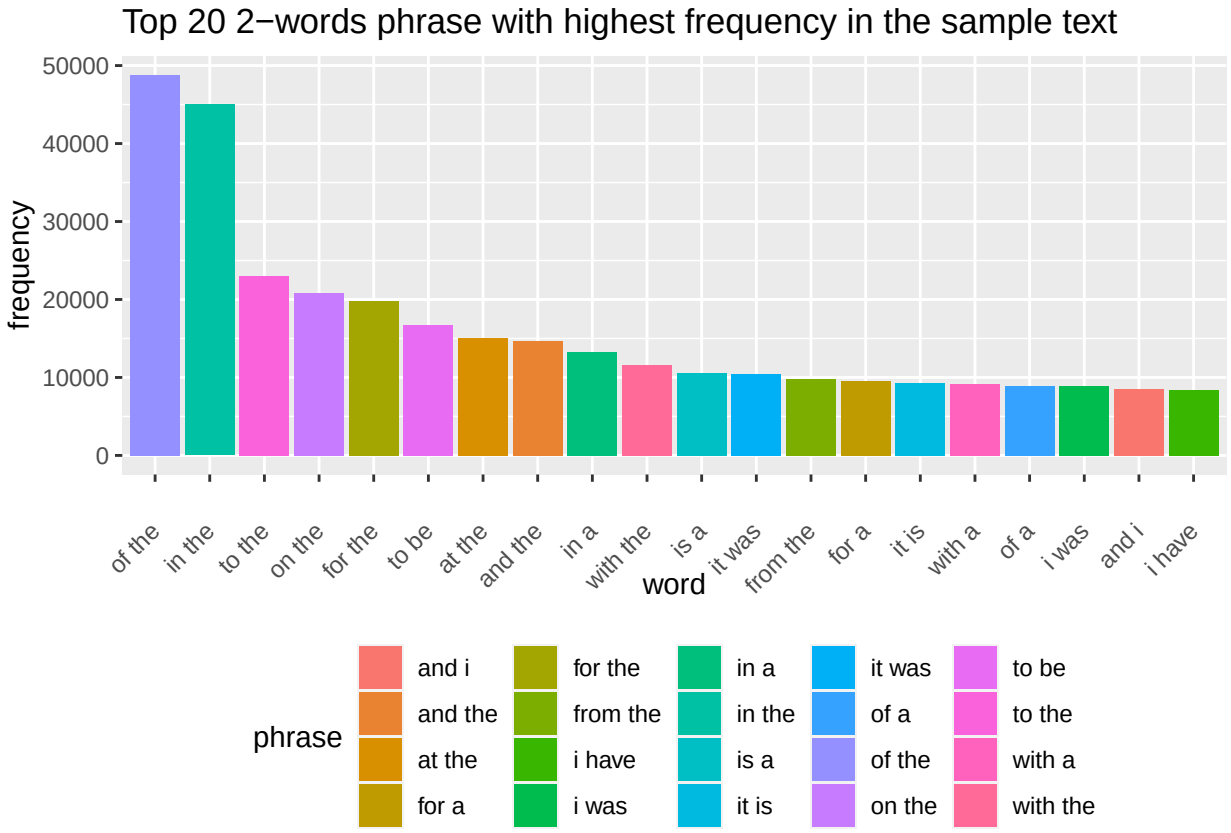
```
biPt <- bigram %>%
    count(bigram, sort = TRUE) %>%
    mutate(prop = n/sum(n)) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    mutate(phrase = paste(word1, word2)) %>%
    na.omit()
```

```
head(biPt, 20)
```

```
##    word1 word2     n         prop   phrase
## 1     of   the 48821 0.0049532932   of the
## 2     in   the 45007 0.0045663314   in the
## 3     to   the 22994 0.0023329310   to the
## 4     on   the 20785 0.0021088097   on the
## 5    for   the 19791 0.0020079602  for the
## 6     to    be 16694 0.0016937440    to be
## 7     at   the 15035 0.0015254248   at the
## 8    and   the 14637 0.0014850444  and the
## 9     in     a 13261 0.0013454378     in a
## 10  with   the 11597 0.0011766113 with the
## 11    is     a 10535 0.0010688627     is a
## 12    it   was 10485 0.0010637897   it was
## 13  from   the  9797 0.0009939865 from the
## 14   for     a  9501 0.0009639548    for a
## 15    it    is  9239 0.0009373728    it is
## 16  with     a  9117 0.0009249949   with a
## 17    of     a  8919 0.0009049061     of a
## 18     i   was  8915 0.0009045003    i was
## 19   and     i  8510 0.0008634097    and i
## 20     i  have  8386 0.0008508289   i have
```

top 20 phrase in bigram

```
g2 <- ggplot(biPt[1:20, ], aes(x = reorder(phrase, -n), y = n,
    fill = phrase))
g2 <- g2 + geom_bar(stat = "identity") + labs(x = "word", y = "frequency",
    title = "Top 20 2-words phrase with highest frequency in the sample text") +
    theme(axis.text.x = element_text(angle = 45, vjust = 0.5,
        hjust = 1), legend.position = "bottom")
g2
```



"of the", "in the", and "to the" have the three highest frequency in the bigrams.

Trigram

```
trigram <- sample_txt %>%
    unnest_tokens(trigram, text, token = "ngrams", n = 3)
```

Trigram phrase table with proportion

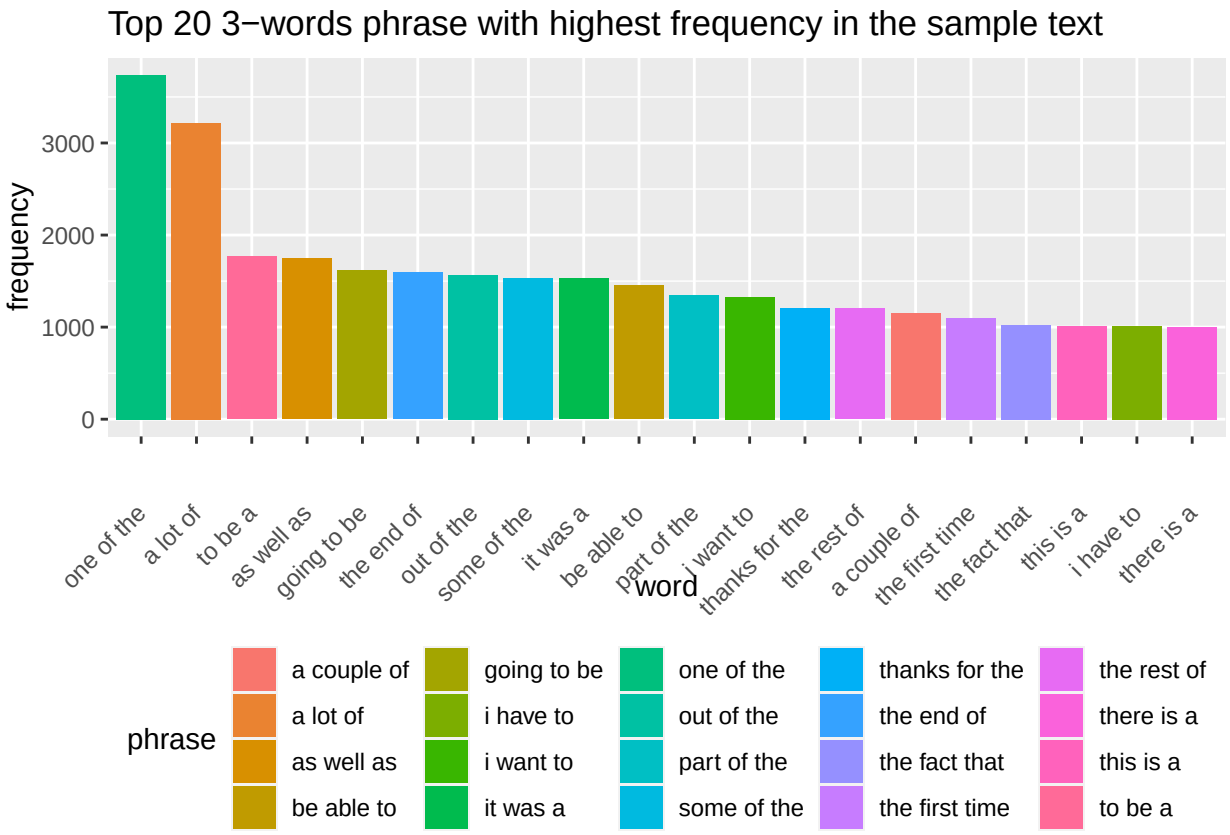
```
triPt <- trigram %>%
    count(trigram, sort = TRUE) %>%
    mutate(prop = n/sum(n)) %>%
    separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
    mutate(phrase = paste(word1, word2, word3)) %>%
    na.omit()
```

```
head(triPt, 20)
```

```
##      word1  word2 word3    n         prop         phrase
## 2      one     of   the 3739 0.0003929319     one of the
## 3        a    lot    of 3218 0.0003381799       a lot of
## 4       to     be     a 1769 0.0001859044        to be a
## 5       as   well    as 1744 0.0001832771     as well as
## 6   going     to    be 1612 0.0001694052    going to be
## 7      the    end    of 1595 0.0001676187     the end of
## 8      out     of   the 1567 0.0001646762     out of the
## 9     some     of   the 1534 0.0001612082    some of the
## 10     it    was     a 1525 0.0001602624       it was a
## 11     be   able    to 1452 0.0001525908     be able to
## 12   part     of   the 1342 0.0001410309    part of the
## 13      i   want    to 1327 0.0001394546      i want to
## 14 thanks    for   the 1206 0.0001267387 thanks for the
## 15    the   rest    of 1203 0.0001264234    the rest of
## 16      a couple    of 1152 0.0001210638    a couple of
## 17    the  first  time 1097 0.0001152838 the first time
## 18    the   fact  that 1022 0.0001074021  the fact that
## 19   this     is     a 1014 0.0001065614      this is a
## 20      i   have    to 1013 0.0001064563      i have to
## 21  there     is     a 1000 0.0001050901     there is a
```

top 20 phrase in trigram

```
g3 <- ggplot(triPt[1:20, ], aes(x = reorder(phrase, -n), y = n,
    fill = phrase))
g3 <- g3 + geom_bar(stat = "identity") + labs(x = "word", y = "frequency",
    title = "Top 20 3-words phrase with highest frequency in the sample text") +
    theme(axis.text.x = element_text(angle = 45, vjust = 0.5,
        hjust = 1), legend.position = "bottom")
g3
```



"one of the", "a lot of", and "to be a" have the three highest frequency in the trigrams.

Save the phrase tables for later predictions

```
saveRDS(uniPt, "./ngramTable/ngram1_phrase_table.rds")
saveRDS(biPt, "./ngramTable/ngram2_phrase_table.rds")
saveRDS(triPt, "./ngramTable/ngram3_phrase_table.rds")
```

How many unique words is needed to cover 50% of the sample text?

```
count = 0
for (i in 1:nrow(uniPt)) {
    count = count + uniPt$n[i]
    if (count >= 0.5 * wdct) {
        break
    }
}
i
```

```
## [1] 156
```

```
i/nrow(uniPt)
```

```
## [1] 0.0008411381
```

It require 156 number of unique words to cover 50% of the sample text, which 8.411381e-04 of the total number of unique words.

Exploratory analysis summary

I think 5% of the original data can already have a accurate representation of the training set since the sample already have 10428127 words.
The reduction of sample set can allow more rapid exploration of the data while keeping the accuracy of the findings.

Text Prediction

Plan

Prediction algorithm

I will create a prediction algorithm base on the n-grams words frequency. The frequency convert to probability.

1. Find all 3-grams phrase that contain the input word
2. Use the frequency of all the phrase to generate a probability distribution to determine which which is the next word.
3. For words that hasn't appears in the n-grams, it will return a random 3 word phrase generated by the frequency (the higher the frequency in the training set, the higher the chance that the phrase is output)

Shiny app

It will have a side panel which allow user to input word. It will also have the main panel which will produce output phrase from the prediction algorithm and the top three most probable phrase base on the probability distribution

Prediction

Model 1

function 1

```
predic1 <- function(sen, n, len) {
    filePath <- paste0("ngramTable/ngram", as.character(n), "_phrase_table.rds")
    pt <- readRDS(filePath)
    if (n == 1) {
        return(pt[1:10, 1])
    }
    if (n == 2) {
        resPt <- pt[pt[, 1] == sen[len], c(3, 5)]
    } else {
        resPt <- pt[which(apply(pt[, 1:(n - 1)], 1, function(x) all(x ==
            sen[(len - n + 2):len]))), c(1:(n + 1), n + 3)]
    }
    if (nrow(resPt) == 0) {
        return(predic1(sen, n - 1, len))
    } else {
        return(resPt[1:10, ] %>%
            na.omit())
    }
}

pred1 <- function(sen) {
    start_time <- Sys.time()
```

```
    sen <- sen %>%
        removePunctuation() %>%
        tolower() %>%
        str_split(" ")
    sen <- sen[[1]]
    len <- length(sen)
    n <- min(len, 3)
    predic1(sen, n, len)
}
```

This function use the unigram, bigram and trigram to predict the next word.
It will return the top 10 most probable result according to the n-grams.
1. It first check if there is any row in the trigram match the last two words in the sentence. If yes, it returns the top 10 highest proportion result. 2. If there is no row in the trigram that matches the sentence, it check from the bigram and return the result if it can find any row that matches. 3. If it still could not find any matches, it will return the top 10 highest frequency word in the unigram.

Testing 1

```
start_time <- Sys.time()
pred1("Well I'm pretty sure my granny has some old bagpipes in her garage I'll dust them off and be on
```

Test 1.1

```
##       word1 word2    word3   n         phrase
## 1111     on    my      way 148      on my way
## 1623     on    my      own 118      on my own
## 3460     on    my     blog  73     on my blog
## 3782     on    my     face  69     on my face
## 4664     on    my     mind  60     on my mind
## 6692     on    my    phone  47    on my phone
## 8807     on    my     list  39     on my list
## 13580    on    my computer  29 on my computer
## 15040    on    my     part  27     on my part
## 15846    on    my birthday  26 on my birthday
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 35.15356 secs.

```
start_time <- Sys.time()
pred1("Talking to your mom has the same effect as a hug and helps reduce your")
```

Test 1.2

```
##          word1 word2    word3 n               phrase
## 571932  reduce  your exposure 3 reduce your exposure
## 1182712 reduce  your   intake 2   reduce your intake
## 4005938 reduce  your   credit 1   reduce your credit
## 4005939 reduce  your     debt 1     reduce your debt
## 4005940 reduce  your   energy 1   reduce your energy
## 4005941 reduce  your  monthly 1  reduce your monthly
## 4005942 reduce  your     risk 1     reduce your risk
## 4005943 reduce  your   team's 1   reduce your team's
## 4005944 reduce  your     word 1     reduce your word
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 1.306526 mins.

```
start_time <- Sys.time()
pred1("Be grateful for the good times and keep the faith during the")
```

Test 1.3

```
##        word1 word2     word3   n              phrase
## 2013  during   the       day 103       during the day
## 2039  during   the     first 102     during the first
## 4428  during   the      week  62      during the week
## 6621  during   the         s  47         during the s
## 8137  during   the      same  41      during the same
## 8400  during   the    summer  40    during the summer
## 8689  during   the      past  39      during the past
## 8998  during   the      last  38      during the last
## 9362  during   the    season  37    during the season
## 11056 during   the recession  33 during the recession
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 1.001756 mins.

Evaluation

Result From the result, we can see that the there is too many result for some testing and the time is not ideal.

Room of improvement we can improve the algorithm in these direction:

1. Increase specificity
2. Decrease processing time

Model 2

I will create quadgram and quintgram to improve accuracy

Create quadgram and quintgram

```
sample_txt <- readLines("sample.txt")
sample_txt <- gsub("http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+",
    "", sample_txt) %>%
    removePunctuation() %>%
    removeNumbers() %>%
    tolower() %>%
    stripWhitespace()
sample_txt <- data.frame(text = sample_txt)

quadgram <- sample_txt %>%
    unnest_tokens(quadgram, text, token = "ngrams", n = 4)
quadPt <- quadgram %>%
    count(quadgram, sort = TRUE) %>%
    mutate(prop = n/sum(n)) %>%
    separate(quadgram, c("word1", "word2", "word3", "word4"),
        sep = " ") %>%
    mutate(phrase = paste(word1, word2, word3, word4)) %>%
    na.omit()
saveRDS(quadPt, "./ngramTable/ngram4_phrase_table.rds")
rm(list = c("quadgram", "quadPt"))

quintgram <- sample_txt %>%
    unnest_tokens(quintgram, text, token = "ngrams", n = 5)
quintPt <- quintgram %>%
    count(quintgram, sort = TRUE) %>%
    mutate(prop = n/sum(n)) %>%
    separate(quintgram, c("word1", "word2", "word3", "word4",
        "word5"), sep = " ") %>%
    mutate(phrase = paste(word1, word2, word3, word4, word5)) %>%
    na.omit()
saveRDS(quintPt, "./ngramTable/ngram5_phrase_table.rds")
rm(list = c("quintgram", "quintPt"))
```

function 2

```
predic2 <- function(sen, n, len) {
    filePath <- paste0("ngramTable/ngram", as.character(n), "_phrase_table.rds")
    pt <- readRDS(filePath)
    if (n == 1) {
        return(pt[1:10, 1])
    }
    if (n == 2) {
        resPt <- pt[pt[, 1] == sen[len], c(3, 5)]
    } else {
        resPt <- pt[which(apply(pt[, 1:(n - 1)], 1, function(x) all(x ==
            sen[(len - n + 2):len]))), c(1:(n + 1), n + 3)]
    }
    if (nrow(resPt) == 0) {
        return(predic2(sen, n - 1, len))
```

```
    } else {
        return(resPt[1:10, ] %>%
            na.omit())
    }
}

pred2 <- function(sen) {
    start_time <- Sys.time()
    sen <- sen %>%
        removePunctuation() %>%
        tolower() %>%
        str_split(" ")
    sen <- sen[[1]]
    len <- length(sen)
    n <- min(len, 5)
    predic2(sen, n, len)
}
```

This function is the same as the previous one other than this one uses n-gram phrase table of higher order (4 and 5).

Testing 2

```
start_time <- Sys.time()
pred2("Well I'm pretty sure my granny has some old bagpipes in her garage I'll dust them off and be on m
```

Test 2.1

```
##         word1 word2 word3 word4 word5 n             phrase
## 1581498   and    be    on    my mouth 1 and be on my mouth
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 1.447844 mins.

```
start_time <- Sys.time()
pred2("Talking to your mom has the same effect as a hug and helps reduce your")
```

Test 2.2

```
##          word1 word2    word3 n               phrase
## 571932  reduce  your exposure 3 reduce your exposure
## 1182712 reduce  your   intake 2   reduce your intake
## 4005938 reduce  your   credit 1   reduce your credit
## 4005939 reduce  your     debt 1     reduce your debt
## 4005940 reduce  your   energy 1   reduce your energy
```

```
## 4005941 reduce  your  monthly 1  reduce your monthly
## 4005942 reduce  your     risk 1     reduce your risk
## 4005943 reduce  your   team's 1   reduce your team's
## 4005944 reduce  your     word 1     reduce your word
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 4.909918 mins.

```
start_time <- Sys.time()
pred2("Be grateful for the good times and keep the faith during the")
```

Test 2.3

```
##         word1  word2 word3   word4 n                   phrase
## 2779917 faith during   the   three 1   faith during the three
## 2779918 faith during   the worship 1 faith during the worship
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 2.550962 mins.

Model 3

Decrease run time and processing powered needed of the function.

function 3

```
predic3 <- function(sen, n, len) {
    filePath <- paste0("ngramTable/ngram", as.character(n), "_phrase_table.rds")
    pt <- readRDS(filePath)
    if (n == 1) {
        return(pt[1:10, 1])
    }
    print(sen[(len - n + 2):len])
    print(n)
    if (n == 2) {
        resPt <- pt[pt[, 1] == sen[len], c(3, 5)]
    } else {
        resPt <- pt
        for (i in 1:(n - 1)) {
            resPt <- resPt[resPt[, i] == sen[len - n + i + 1],
                ]
        }
    }
    if (nrow(resPt) == 0) {
        return(predic3(sen, n - 1, len))
```

```
    } else {
        return(resPt[1:10, ] %>%
            na.omit())
    }
}

pred3 <- function(sen) {
    start_time <- Sys.time()
    sen <- sen %>%
        removePunctuation() %>%
        tolower() %>%
        str_split(" ")
    sen <- sen[[1]]
    len <- length(sen)
    n <- min(len, 5)
    res <- predic3(sen, n, len)
    print(Sys.time() - start_time)
    res
}
```

Testing 3

```
start_time <- Sys.time()
pred3("Well I'm pretty sure my granny has some old bagpipes in her garage I'll dust them off and be on m
```

Test 3.1

```
## [1] "and" "be"  "on"  "my"
## [1] 5
## Time difference of 19.67073 secs
```

```
##         word1 word2 word3 word4 word5 n         prop             phrase
## 1581498   and    be    on    my mouth 1 1.127204e-07 and be on my mouth
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 19.6746 secs.

```
start_time <- Sys.time()
pred3("Talking to your mom has the same effect as a hug and helps reduce your")
```

Test 3.2

```
## [1] "and"    "helps"  "reduce" "your"
## [1] 5
## [1] "helps"  "reduce" "your"
```

```
## [1] 4
## [1] "reduce" "your"
## [1] 3
## Time difference of 46.48669 secs
```

```
##          word1 word2    word3 n         prop               phrase
## 571932  reduce  your exposure 3 3.152703e-07 reduce your exposure
## 1182712 reduce  your   intake 2 2.101802e-07   reduce your intake
## 4005938 reduce  your   credit 1 1.050901e-07   reduce your credit
## 4005939 reduce  your     debt 1 1.050901e-07     reduce your debt
## 4005940 reduce  your   energy 1 1.050901e-07   reduce your energy
## 4005941 reduce  your  monthly 1 1.050901e-07  reduce your monthly
## 4005942 reduce  your     risk 1 1.050901e-07     reduce your risk
## 4005943 reduce  your   team's 1 1.050901e-07   reduce your team's
## 4005944 reduce  your     word 1 1.050901e-07     reduce your word
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 46.49199 secs.

```
start_time <- Sys.time()
pred3("Be grateful for the good times and keep the faith during the")
```

Test 3.3

```
## [1] "the"    "faith"  "during" "the"
## [1] 5
## [1] "faith"  "during" "the"
## [1] 4
## Time difference of 41.61286 secs
```

```
##         word1  word2 word3   word4 n        prop                   phrase
## 2779917 faith during   the   three 1 1.08845e-07   faith during the three
## 2779918 faith during   the worship 1 1.08845e-07 faith during the worship
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 41.62364 secs.

Evaluation

Result The time taken for each test has significantly decreased even with the quadgram and quintgram. After testing, I found out that the sentence almost never matches the phrase in quintgram.

Room of improvement Remove quintgram from the prediction function

Model 4

Decrease run time and processing powered needed of the function.

function 4

```
predic4 <- function(sen, n, len) {
    filePath <- paste0("ngramTable/ngram", as.character(n), "_phrase_table.rds")
    pt <- readRDS(filePath)
    if (n == 1) {
        return(pt[1:10, 1])
    }
    print(sen[(len - n + 2):len])
    print(n)
    if (n == 2) {
        resPt <- pt[pt[, 1] == sen[len], c(3, 5)]
    } else {
        resPt <- pt
        for (i in 1:(n - 1)) {
            resPt <- resPt[resPt[, i] == sen[len - n + i + 1],
                ]
        }
    }
    if (nrow(resPt) == 0) {
        return(predic4(sen, n - 1, len))
    } else {
        return(resPt[1:10, ] %>%
            na.omit())
    }
}

pred4 <- function(sen) {
    start_time <- Sys.time()
    sen <- sen %>%
        removePunctuation() %>%
        tolower() %>%
        str_split(" ")
    sen <- sen[[1]]
    len <- length(sen)
    n <- min(len, 4)
    res <- predic4(sen, n, len)
    print(Sys.time() - start_time)
    res
}
```

Testing 4

```
start_time <- Sys.time()
pred4("Well I'm pretty sure my granny has some old bagpipes in her garage I'll dust them off and be on
```

Test 4.1

```
## [1] "be" "on" "my"
## [1] 4
## Time difference of 16.83587 secs
```

```
##         word1 word2 word3 word4 n         prop         phrase
## 658283     be    on    my  feet 2 2.176901e-07  be on my feet
## 658284     be    on    my radio 2 2.176901e-07 be on my radio
## 658285     be    on    my   way 2 2.176901e-07   be on my way
## 2020956    be    on    my     a 1 1.088450e-07     be on my a
## 2020957    be    on    my   bus 1 1.088450e-07   be on my bus
## 2020958    be    on    my  ipod 1 1.088450e-07  be on my ipod
## 2020959    be    on    my level 1 1.088450e-07 be on my level
## 2020960    be    on    my  list 1 1.088450e-07  be on my list
## 2020961    be    on    my  mind 1 1.088450e-07  be on my mind
## 2020962    be    on    my mouth 1 1.088450e-07 be on my mouth
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 16.83947 secs.

```
start_time <- Sys.time()
pred4("Talking to your mom has the same effect as a hug and helps reduce your")
```

Test 4.2

```
## [1] "helps"  "reduce" "your"
## [1] 4
## [1] "reduce" "your"
## [1] 3
## Time difference of 32.0298 secs
```

```
##          word1 word2    word3 n         prop               phrase
## 571932  reduce  your exposure 3 3.152703e-07 reduce your exposure
## 1182712 reduce  your   intake 2 2.101802e-07   reduce your intake
## 4005938 reduce  your   credit 1 1.050901e-07   reduce your credit
## 4005939 reduce  your     debt 1 1.050901e-07     reduce your debt
## 4005940 reduce  your   energy 1 1.050901e-07   reduce your energy
## 4005941 reduce  your  monthly 1 1.050901e-07  reduce your monthly
## 4005942 reduce  your     risk 1 1.050901e-07     reduce your risk
## 4005943 reduce  your   team's 1 1.050901e-07   reduce your team's
## 4005944 reduce  your     word 1 1.050901e-07     reduce your word
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 32.0345 secs.

```
start_time <- Sys.time()
pred4("Be grateful for the good times and keep the faith during the")
```

Test 4.3

```
## [1] "faith"  "during" "the"
## [1] 4
## Time difference of 21.3866 secs

##         word1  word2 word3   word4 n        prop                    phrase
## 2779917 faith during   the   three 1 1.08845e-07   faith during the three
## 2779918 faith during   the worship 1 1.08845e-07 faith during the worship
```

```
time_diff <- Sys.time() - start_time
```

The time taken to produce the result is 21.39751 secs.

Generate Text

This function will generate a sentence with n number words.
1. It generates the first word at random with all words having equal weights. 2. It generates the second word by matching the first word with the bigram and takes the phrase with highest frequency. 3. It generates the remaining words by matching the last two words with the trigram and takes the phrase with highest frequency (There is a probability of 0.2 that the next word will be taken a random with weight corresponding to the frequency in the sample text file.) (If it cannot match the words, it find it in the n-gram of lower order)

function

```
#' The function export is generate(n) where n is the number of words in the sentence it generate.
#' This function will generate word using the word that is most frequent with a probability of 0.2 wher
#' ' done it in beach their of at i in isone was a siren rape and hear destroy ttt trys in'
library(stringr)



#' sen is the vector of words that needs to be matched
#' pt is the phrase table
#' n is the degree of the phrase table
match_phrase <- function(sen, pt, n) {
    sen <- str_split(sen, " ")[[1]]
    len = length(sen)
    resPt <- pt

    for (i in 1:(n - 1)) {
        resPt <- resPt[resPt[, i] == sen[len - n + i + 1], ]
    }
```

```
    resPt
}

roll <- function() {
    if (sample(0:1, size = 1, prob = c(0.2, 0.8)) == 1) {
        return(TRUE)
    } else {
        return(FALSE)
    }

}

genFirst2 <- function() {

    # generate first word
    pt1 <- readRDS("./ngramTable/ngram1_phrase_table.rds")

    word <- sample(pt1$word, size = 1, prob = pt1$n)
    sen <- word
    rm(list = "pt1")

    # generate second word
    pt2 <- readRDS("./ngramTable/ngram2_phrase_table.rds")
    respt <- match_phrase(sen, pt2, 2)
    if (nrow(respt) == 0) {
        respt <- readRDS("./ngramTable/ngram1_phrase_table.rds")
        word <- sample(respt$word, size = 1, prob = respt$n)
    } else if (roll()) {
        word <- respt$word2[1]
    } else {
        word <- sample(respt$word2, size = 1, prob = respt$n)
    }
    sen <- paste(sen, word)
    rm(list = c("respt", "pt2", "word"))

    return(sen)
}

genNext <- function(sen, pt3) {
    respt <- match_phrase(sen, pt3, 3)
    if (nrow(respt) == 0) {
        pt2 <- readRDS("./ngramTable/ngram2_phrase_table.rds")
        respt <- match_phrase(sen, pt2, 2)
        if (nrow(respt) == 0) {
            respt <- readRDS("./ngramTable/ngram1_phrase_table.rds")
            word <- sample(respt$word, size = 1, prob = respt$n)
            rm(list = "respt")
        } else if (roll()) {
            word <- respt$word2[1]
        } else {
            word <- sample(respt$word2, size = 1, prob = respt$n)
            rm(list = "respt")
        }
```

```
        rm(list = "pt2")
    } else if (roll()) {
        word <- respt$word3[1]
    } else {
        word <- sample(respt$word3, size = 1, prob = respt$n)
    }
    sen <- paste(sen, word)
    return(sen)
}
```

```
generate <- function(n) {
    n <- as.numeric(n)
    if (is.na(n)) {
        return("Please input a positive integer")
    }
    if (n < 0 | n > 30 | !n%%1 == 0) {
        return("Number not valid. (need to be positive integer smaller than or equal to 30)")
    }
    if (n == 1) {
        pt1 <- readRDS("./ngramTable/ngram1_phrase_table.rds")
        word <- sample(pt1$word, size = 1)
        return(word)
    } else if (n == 2) {
        return(genFirst2())
    } else if (n > 2) {
        pt3 <- readRDS("./ngramTable/ngram3_phrase_table.rds")
        sen <- genFirst2()
        for (i in 1:(n - 2)) {
            sen <- genNext(sen, pt3)
        }
        return(sen)
    }
}
```

Test

Test 1 (generate 30 words sentence)

```
start_time <- Sys.time()
generate(30)
```

```
## [1] "it was a totally different dynamic dont know what i was a little bit for not being able to be ja
```

```
time_diff <- Sys.time() - start_time
```

The time difference is 22.58152 secs

Test 2 (generate 30 words sentence)

```
start_time <- Sys.time()
generate(30)
```

```
## [1] "suggest that another friend named fun fun was had by all and i have to wait for your support an
```

```
time_diff <- Sys.time() - start_time
```

The time difference is 17.55224 secs

Invalid input test

```
generate(-1)
```

```
## [1] "Number not valid. (need to be positive integer smaller than or equal to 30)"
```

```
generate(0.1)
```

```
## [1] "Number not valid. (need to be positive integer smaller than or equal to 30)"
```

```
generate("ABC")
```

```
## Warning in generate("ABC"): NAs introduced by coercion
```

```
## [1] "Please input a positive integer"
```