Data Science Capstone - Quiz 2

Natural language processing

For how to develop text input prediction models, refer to this report http://rpubs.com/Nov05/459931, in which only Twitter text was explored. However, for actual models, I might use all the English text files, e.g. Twitter, blogs and news, or at least a sample of text of all the files. Also, in the coming weeks, for the actual product I will reduce the N-gram dictionary size to improve the performance, or use lemmatization and/or stemming to push it further.

N-gram Modeling With Markov Chains
https://sookocheff.com/post/nlp/ngram-modeling-with-markov-chains/

```
library(stringr)
library(tm)
library(ggplot2)
library(ngram)

# constants
co_twitter_en = "../data/capstone/en_US/en_US.twitter.txt"
co_blogs_en = "../data/capstone/en_US/en_US.blogs.txt"
co_news_en = "../data/capstone/en_US/en_US.news.txt"

co_text_attr_en = "../data/capstone/text_attr_en.rds"

co_tidy_twitter_en = "../data/capstone/tidy_twitter_en.rds"
co_tidy_nostop_twitter_en = "../data/capstone/tidy_nostop_twitter_en.rds"
co_tidy_blogs_en = "../data/capstone/tidy_blogs_en.rds"
co_tidy_news_en = "../data/capstone/tidy_news_en.rds"

co_3gram_en = "../data/capstone/3gram_en.rds"
co_1gram_twitter_en = "../data/capstone/1gram_twitter_en.rds"
co_2gram_twitter_en = "../data/capstone/2gram_twitter_en.rds"
co_3gram_twitter_en = "../data/capstone/3gram_twitter_en.rds"
co_1gram_nostop_twitter_en = "../data/capstone/1gram_nostop_twitter_en.rds"
co_2gram_nostop_twitter_en = "../data/capstone/2gram_nostop_twitter_en.rds"
co_3gram_nostop_twitter_en = "../data/capstone/3gram_nostop_twitter_en.rds"
```

Here is the code block to get 3-grams from all the English texts.

```
tidyText <- function(file, tidyfile) {
  con <- file(file, open="r")
```

```
  lines <- readLines(con)
  close(con)

  lines <- tolower(lines)
  # split at all ".", "," and etc.
  lines <- unlist(strsplit(lines, "[.,:;!?(){}<>]+")) # 5398319 lines

  # replace all non-alphanumeric characters with a space at the beginning/end of a word.
  lines <- gsub("^[^a-z0-9]+|[^a-z0-9]+$", " ", lines) # at the begining/end of a line
  lines <- gsub("[^a-z0-9]+\\s", " ", lines) # before space
  lines <- gsub("\\s[^a-z0-9]+", " ", lines) # after space
  lines <- gsub("\\s+", " ", lines) # remove mutiple spaces
  lines <- str_trim(lines) # remove spaces at the beginning/end of the line

  saveRDS(lines, file=tidyfile)
}

tidyText(co_news_en, co_tidy_news_en)
tidyText(co_blogs_en, co_tidy_blogs_en)

df_news <- readRDS(co_tidy_news_en)
df_blogs <- readRDS(co_tidy_blogs_en)
df_twitter <- readRDS(co_tidy_twitter_en)
lines <- c(df_news, df_blogs, df_twitter)
rm(df_news, df_blogs, df_twitter)

# remove lines that contain less than 3 words, or ngram() would throw errors.
lines <- lines[str_count(lines, "\\s+")>1] # reduce 10483160 elements to 7730009 elements
# this line took long time
trigram <- ngram(lines, n=3); rm(lines)
# this line took long time
df <- get.phrasetable(trigram); rm(trigram)
saveRDS(df, co_3gram_en)
```

For each of the sentence fragments below use your natural language processing algorithm to predict the next word in the sentence.

1. The guy in front of me just bought a pound of bacon, a bouquet, and a case of

Options: prezels, soda, beer, cheese

Answer: beer

```
df <- readRDS(co_3gram_en)
head(df[grep("^case of", df[,1]),], 10)
```

```
##                    ngrams freq         prop
## 5166          case of the   385 7.550852e-06
## 25725           case of a   116 2.275062e-06
## 116350         case of an    33 6.472158e-07
## 164531       case of beer    24 4.707024e-07
## 179991       case of this    22 4.314772e-07
## 219509 case of emergency    19 3.726394e-07
## 298460         case of my    14 2.745764e-07
## 346130        case of any    13 2.549638e-07
## 384479        case of one    11 2.157386e-07
## 475693         case of it     9 1.765134e-07
```

2. You're the reason why I smile everyday. Can you follow me please? It would mean the

Options: world, best, most, universe

Answer: world

```
head(df[grep("^mean the ", df[,1]),], 10)
```

```
##                      ngrams freq         prop
## 7337         mean the world   298 5.844555e-06
## 134767        mean the same    29 5.687654e-07
## 182897        mean the most    22 4.314772e-07
## 188910 mean the difference    21 4.118646e-07
## 244893       mean the whole    17 3.334142e-07
## 317025         mean the one    14 2.745764e-07
## 390432         mean the end    11 2.157386e-07
## 720089       mean the other     6 1.176756e-07
## 865698        mean the rest     5 9.806301e-08
## 880671      mean the person     5 9.806301e-08
```

3. Hey sunshine, can you follow me and make me the

Options: bluest, smelliest, saddest, happiest

Answer: happiest

Note: The top frequence of the 3-grams is “me the f*ck“. Interesting. Probably need to add the f-word to stop word list? Lol.

```
rbind(df[grep("^me the bluest", df[,1]),],
      df[grep("^me the smelliest", df[,1]),],
      df[grep("^me the saddest", df[,1]),],
      df[grep("^me the happiest", df[,1]),])
```

```
##                  ngrams freq         prop
## 89105 me the happiest    41 8.041167e-07
```

4. Very early observations on the Bills game: Offence still struggling but the

Options: crowd, defense, referees, players(wrong)

Answer: defense

Note: It didn’t match anything in the options by using Twitter N-gram dictionary. Probably need to use models generated by all the English files.

```
head(df[grep("^struggling but", df[,1]),], 10) # It didn't match anything in the options.
```

```
##                              ngrams freq        prop
## 5138208   struggling but avoiding     1 1.96126e-08
## 5346661  struggling but westbrook     1 1.96126e-08
## 16775419  struggling but remember     1 1.96126e-08
```

```
str <- "Very early observations on the Bills game: Offence still struggling but the"
str <- removeWords(str, stopwords("en")); str <- gsub("\\s+", " ", str); str
```

```
## [1] "Very early observations Bills game: Offence still struggling "
```

```
head(df[grep("^still struggling", df[,1]),], 10)
```

```
##                            ngrams freq         prop
## 120290     still struggling with    32 6.276032e-07
## 166603       still struggling to    24 4.707024e-07
## 1490167       still struggling a     3 5.883780e-08
## 2493850 still struggling through     2 3.922520e-08
## 5869312    still struggling find     1 1.961260e-08
## 6914234    still struggling then     1 1.961260e-08
## 7123652    still struggling over     1 1.961260e-08
## 8925201   still struggling after     1 1.961260e-08
## 9023578     still struggling but     1 1.961260e-08
## 9377036  still struggling uphill     1 1.961260e-08
```

```
rbind(df[grep("^still struggling crowd", df[,1]),],
      df[grep("^still struggling defense", df[,1]),],
```

```
      df[grep("^still struggling referees", df[,1]),],
      df[grep("^still struggling players", df[,1]),])
## [1] ngrams freq   prop
## <0 rows> (or 0-length row.names)
```

5. Go on a romantic date at the

Options: mall, grocery(wrong), movies, beach

Answer: beach

Note: Should consider sentiments?

```
rbind(df[grep("^date at mall", df[,1]),],
      df[grep("^date at grocery", df[,1]),],
      df[grep("^date at movie", df[,1]),],
      df[grep("^date at beach", df[,1]),])
## [1] ngrams freq   prop
## <0 rows> (or 0-length row.names)
head(df[grep("romantic date", df[,1]),],10)
##                         ngrams freq        prop
## 1935075      a romantic date     3 5.88378e-08
## 4620944  other romantic date     1 1.96126e-08
## 5630589  romantic date stuff     1 1.96126e-08
## 7282670     romantic date as     1 1.96126e-08
## 7717774   romantic date with     1 1.96126e-08
## 8259964     a bromantic date     1 1.96126e-08
## 9292748   bromantic date wid     1 1.96126e-08
## 9662111    hot romantic date     1 1.96126e-08
## 12053230  romantic dates but     1 1.96126e-08
## 14294763  the romantic dates     1 1.96126e-08
```

6. Well I'm pretty sure my granny has some old bagpipes in her garage I'll dust them o□ and be on my

Options: way, horse, motorcycle, phone

Answer: way

```
head(df[grep("^on my ", df[,1]),], 10)
##               ngrams freq         prop
## 271       on my way  2334 4.577581e-05
## 1498      on my own   904 1.772979e-05
## 1518     on my mind   897 1.759250e-05
## 2004    on my phone   745 1.461139e-05
## 2237     on my blog   689 1.351308e-05
```

```
## 2507     on my face   640 1.255206e-05
## 4194     on my list   446 8.747220e-06
## 7654 on my computer   288 5.648429e-06
## 9068 on my birthday   256 5.020826e-06
## 9170     on my ipod   254 4.981601e-06
```

7. Ohhhhh #PointBreak is on tomorrow. Love that film and haven't seen it in quite some

Options: thing, weeks, time, years

Answer: time

```
head(df[grep("^quite some ", df[,1]),], 10)
##                       ngrams freq         prop
## 7053        quite some time   307 6.021069e-06
## 2211193   quite some people     2 3.922520e-08
## 2531118  quite some company     2 3.922520e-08
## 2752442      quite some way     2 3.922520e-08
## 3607237 quite some distance     2 3.922520e-08
## 3911224    quite some years     2 3.922520e-08
## 6350210   quite some months     1 1.961260e-08
## 6586907       quite some cv     1 1.961260e-08
## 8383393      quite some fun     1 1.961260e-08
## 8801999     quite some news     1 1.961260e-08
```

8. After the ice bucket challenge Louis will push his long wet hair out of his eyes with his little

Options: fingers, eyes, ears, toes

Answer: fingers

```
head(df[grep("^his little ", df[,1]),], 10)
##                     ngrams freq         prop
## 178705 his little brother    23 4.510898e-07
## 207115  his little sister    20 3.922520e-07
## 278102    his little girl    15 2.941890e-07
## 410831    his little head    11 2.157386e-07
## 585483    his little body     8 1.569008e-07
## 762139   his little heart     6 1.176756e-07
## 762449   his little hands     6 1.176756e-07
## 773963    his little legs     6 1.176756e-07
## 885943    his little feet     5 9.806301e-08
## 886577    his little face     5 9.806301e-08
rbind(df[grep("his little finger", df[,1]),],
```

```
      df[grep("his little eye", df[,1]),],
      df[grep("his little ear", df[,1]),],
      df[grep("his little toe", df[,1]),])
```

```
##                          ngrams freq         prop
## 1007586      his little finger     5 9.806301e-08
## 1364792     his little fingers     4 7.845041e-08
## 7915789  his little fingernail     1 1.961260e-08
## 10672091    this little finger     1 1.961260e-08
## 1461210        his little eyes     3 5.883780e-08
## 5422883        this little eye     1 1.961260e-08
## 14130296       his little ears     1 1.961260e-08
```

9. Be grateful for the good times and keep the faith during the

Options: worse, bad, hard, sad

Answer: bad

```
head(df[grep("^during the ", df[,1]),], 10)
```

```
##                   ngrams freq         prop
## 2149     during the day   709 1.390533e-05
## 5883    during the week   351 6.884023e-06
## 10322 during the summer   233 4.569736e-06
## 11086  during the first   221 4.334385e-06
## 12966   during the last   197 3.863682e-06
## 22528  during the night   129 2.530026e-06
## 25385  during the month   117 2.294674e-06
## 26888   during the game   112 2.196611e-06
## 28124   during the time   108 2.118161e-06
## 29095 during the course   105 2.059323e-06
```

```
rbind(df[grep("during the worse", df[,1]),],
      df[grep("during the bad", df[,1]),],
      df[grep("during the hard", df[,1]),],
      df[grep("during the sad", df[,1]),])
```

```
##                       ngrams freq        prop
## 2280517      during the bad     2 3.92252e-08
## 18055748 during the badgers     1 1.96126e-08
## 16068626 during the hardest     1 1.96126e-08
## 23145734    during the hard     1 1.96126e-08
```

10. If this isn't the cutest thing you've ever seen, then you must be

Options: asleep, insensitive, callous, insane

Answer: insane

```
head(df[grep("^must be ", df[,1]),], 10)
```

```
##                   ngrams freq         prop
## 2387          must be a   658 1.290509e-05
## 6429        must be the   328 6.432933e-06
## 18520        must be in   150 2.941890e-06
## 25597      must be done   116 2.275062e-06
## 25839      must be able   115 2.255449e-06
## 29227        must be so   105 2.059323e-06
## 30629        must be an   100 1.961260e-06
## 38048 must be something    84 1.647459e-06
## 38492      must be some    83 1.627846e-06
## 38679      must be nice    83 1.627846e-06
```

```
rbind(df[grep("must be asleep", df[,1]),],
      df[grep("must be insensitive", df[,1]),],
      df[grep("must be callous", df[,1]),],
      df[grep("must be insane", df[,1]),])
```

```
##                  ngrams freq         prop
## 4051010 must be asleep     2 3.922520e-08
## 650541  must be insane     7 1.372882e-07
```

```
str <- "If this isn't the cutest thing you've ever seen, then you must be"
str <- removeWords(str, stopwords("en")); str <- gsub("\\s+", " ", str); str
```

```
## [1] "If cutest thing ever seen, must "
```

Refence report:
http://rstudio-pubs-static.s3.amazonaws.com/387645_d494b67fb45e4d3792fb679eb274291c.html
https://rpubs.com/redneckz/smart-keyboard-basic-modeling