Project 4

Jawaid Hakim

2022-10-18

Contents

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readtext)
library(quanteda)
```

```
## Package version: 3.2.3
## Unicode version: 14.0
## ICU version: 70.1
## Parallel computing: 8 of 8 threads used.
## See https://quanteda.io for tutorials and examples.
```

```
library(quanteda.textplots)
library(quanteda.textmodels)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(1234)
```

```
sms <- read.delim("SMSSpamCollection.txt",
                  sep = '\t',
                  col.names = c('cat', 'sms'),
                  quote = "")
```

```r
corp_sms <- corpus(sms,
                   text_field = "sms")
```

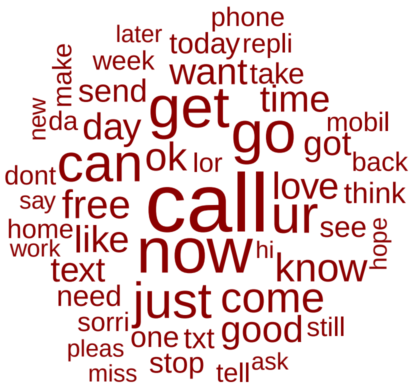
```r
# Add numeric document id, used later to partition train/test sets
corp_sms$id_numeric <- 1:ndoc(corp_sms)

# tokenize
toks_sms <- tokens(corp_sms,
                remove_punct = TRUE,
                remove_numbers = TRUE,
                remove_symbols = TRUE,
                split_hyphens = TRUE) %>%
            tokens_remove(pattern = c(stopwords("en"), "lt", "gt"),
                        valuetype = "fixed",
                        padding = FALSE,
                        min_nchar = 2) %>%
            tokens_tolower(keep_acronyms = TRUE) %>%
            tokens_wordstem()

# doc frequency matrix
dfm_sms <- dfm(toks_sms)
dfm_sms <- dfm_trim(dfm_sms,
                    min_termfreq = 50)
topfeatures(dfm_sms)
```

```
## call  now   go  get  can   ur just come   ok free
##  672  494  451  450  408  391  375  303  285  283
```

```r
textplot_wordcloud(dfm_sms,
                   max_words = 50,
                   rotation = 0.1,
                   color = "darkred")
```



```
fcm_sms <- fcm(dfm_sms,
               context = "document",
               count = "frequency",
               window = 5L)

topfeatures(fcm_sms,
            n = 20,
            scheme = "docfreq")
```

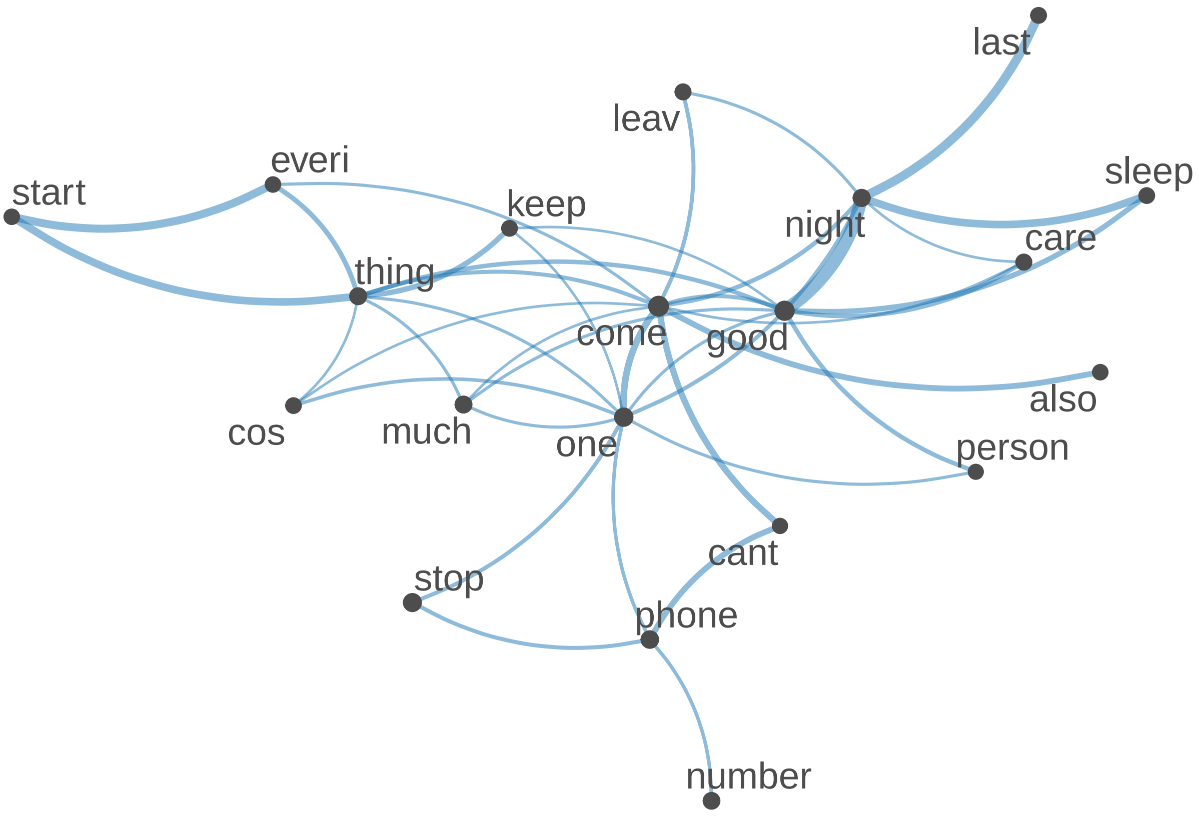
```
##  phone    keep    care     one  number   thing   everi    last    come   night    also
##    138     123     118     116     116     115     112     111     110     108     108
##   cant    stop     cos   sleep   start    leav    good    much  person
##    108     106     106     106     105     105     101     101     101
```

```
feat <- names(topfeatures(fcm_sms,
                          n = 20,
                          scheme = "docfreq"))

fcm_sms_select = fcm_select(fcm_sms,
                            pattern = feat,
                            selection = "keep")

size <- log(colSums(dfm_select(dfm_sms,
                               feat,
                               selection = "keep")))
```

```
textplot_network(fcm_sms_select,
                 min_freq = 0.8,
                 vertex_size = size / max(size) * 3)
```



```
corp_sms_sz <- length(docnames(corp_sms))

# train with 75% data (25% reserved for testing)
id_train <- sample(1:corp_sms_sz, corp_sms_sz * 0.75, replace = FALSE)
head(id_train, 10)
```

```
##  [1] 3715 3085 1097 5379 5205 1057 1916  357  998 2015
```

```
dfm_training <- dfm_subset(dfm_sms, id_numeric %in% id_train)
dfm_test <- dfm_subset(dfm_sms, !id_numeric %in% id_train)
```

```
# Naive Bayes classifier for texts
tmod_nb <- textmodel_nb(dfm_training,
                        dfm_training$cat,
                        prior = "docfreq") # uniform, docfreq, termfreq
summary(tmod_nb)
```

```
##
## Call:
```

```
## textmodel_nb.dfm(x = dfm_training, y = dfm_training$cat, prior = "docfreq")
##
## Class Priors:
## (showing first 2 elements)
##    ham   spam
## 0.8679 0.1321
##
## Estimated Feature Scores:
##            go    great      got       wat        ok     free      win      may
## ham  0.024890 0.006343 0.014532 0.0061020 0.0169410 0.003774 0.001124 0.003051
## spam 0.006862 0.002196 0.001921 0.0002745 0.0008235 0.048861 0.013450 0.002196
##          text      txt       dun       say   alreadi    think     live
## ham  0.004496 0.001044 0.0030510 0.0081895 0.0057005 0.009073 0.001525
## spam 0.029371 0.034038 0.0002745 0.0002745 0.0002745 0.003019 0.006313
##         around      hey     week     now     word     back     like    still
## ham  0.0039342 0.006263 0.004978 0.01895 0.002248 0.007306 0.015335 0.009153
## spam 0.0008235 0.000549 0.019215 0.04008 0.003568 0.005490 0.002745 0.001372
##          send     even       per   friend    custom     prize     claim
## ham  0.008591 0.005299 0.0007226 0.005379 0.0008029 8.029e-05 8.029e-05
## spam 0.014548 0.001647 0.0096075 0.003568 0.0107055 1.812e-02 2.361e-02
```

```
dfm_matched <- dfm_match(dfm_test, features = featnames(dfm_training))
```

```
actual_class <- dfm_matched$cat
predicted_class <- predict(tmod_nb, newdata = dfm_matched)
tab_class <- table(actual_class, predicted_class)
tab_class
```

```
##             predicted_class
## actual_class  ham spam
##         ham  1177   22
##         spam   30  165
```

Sensitivity (True positive rate) = (True Positive)/(True Positive + False Negative) Specificity = (True Negative)/(True Negative + False Positive)

```
confusionMatrix(tab_class, mode = "everything", positive = "spam")
```

```
## Confusion Matrix and Statistics
##
##             predicted_class
## actual_class  ham spam
##         ham  1177   22
##         spam   30  165
##
##                 Accuracy : 0.9627
##                   95% CI : (0.9514, 0.972)
##      No Information Rate : 0.8659
##      P-Value [Acc > NIR] : <2e-16
##
##                    Kappa : 0.8423
##
```

```
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.8824
##             Specificity : 0.9751
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.9817
##               Precision : 0.8462
##                  Recall : 0.8824
##                      F1 : 0.8639
##              Prevalence : 0.1341
##          Detection Rate : 0.1184
##    Detection Prevalence : 0.1399
##       Balanced Accuracy : 0.9287
##
##        'Positive' Class : spam
##
```