Dictionaries and an example of a supervised classifier

10 May 2019

This document describes dictionary methods and supervised machine learning methods for classifying UK prime minister speeches. Let's take a look a set of UK prime minister speeches from the EUSpeech dataset. NB: Use setwd() to set the working directory to the folder that contains English speeches in the file speeches_uk.csv.

```
Sys.setlocale(locale = "en_US.UTF-8")
```

```
## [1] "en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8"
```

```
#load libraries
library(quanteda)
library(stringr)

#read in speeches
speeches <- read.csv(file = "speeches_uk.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE,
                     sep = ",",
                     encoding = "UTF-8")

#remove html tags
speeches$text <- str_replace_all(speeches$text, "<.*?>", "")
#replace multiple white spaces with single white spaces
speeches$text <- str_replace_all(speeches$text, "  ", " ")

#construct a corpus
corpus <- corpus(speeches)
#focus on Cameron and Brown
corpus <- corpus_subset(corpus, speaker != "T. Blair")
#turn the date variable in a date format instead of character format
docvars(corpus, "date") <- as.Date(docvars(corpus, "date"), "%d-%m-%Y")

#turn this into a dfm
corpus.dfm <- dfm(corpus, stem = FALSE,
                  remove=stopwords("english"),
                  remove_punct=TRUE)
dim(corpus.dfm)
```

```
## [1]   776 43954
```

```
#trim this corpus to speed up calculations
corpus.dfm = dfm_trim(corpus.dfm, min_docfreq = 20)
dim(corpus.dfm)
```

```
## [1]  776 3907
```

Dictionary methods

When working with your own dictionary, most of the work will go into evaluating its validity and reliability in order to make sure that it captures the construct that you are looking for. However, once you have settled on a dictionary, it is very easy in quanteda to apply it to a corpus.

Let's say where are interested in how often Cameron and Brown mention immigration, refugees and asylum:

```
#create a dictionary
practice.dict <- dictionary(list(Immigration = c("immi*"),
                                 Refugees = c("refug*"),
                                 Asylum = c("asyl*")))

practice.dict.dfm <- dfm(corpus.dfm, dictionary = practice.dict)

dim(practice.dict.dfm)
```

```
## [1] 776   3
```

```
head(practice.dict.dfm)
```

```
## Document-feature matrix of: 6 documents, 3 features (88.9% sparse).
## 6 x 3 sparse Matrix of class "dfm"
##        features
## docs    Immigration Refugees Asylum
##   text1           0        3      0
##   text2           0        0      0
##   text3           0        0      0
##   text4           0        0      0
##   text5           0        1      0
##   text6           0        0      0
```

As you can see, practice.dict.dfm is a dfm object that contains for every document the number of times each dictionary word appears.

Quanteda contains a number of off-the-shelf dictionaries. Let's take a look at the Lexicoder Sentiment Dictionary from Young and Soroka (2012). It's stored in quanteda as a dictionary object under data_dictionary_LSD2015. Let's apply it the Cameron speeches:

```
#create Cameron corpus and dfm
corpus.cameron <- corpus_subset(corpus, speaker == "D. Cameron")

corpus.cameron.dfm <- dfm(corpus.cameron,
                          stem = FALSE,
                          remove=stopwords("english"),
                          remove_punct=TRUE)

corpus.cameron.dfm = dfm_trim(corpus.cameron.dfm, min_docfreq = 20)

#look at the Lexicoder Sentiment Dictionary in quanteda
str(data_dictionary_LSD2015)
```

```
## Formal class 'dictionary2' [package "quanteda"] with 2 slots
##   ..@ .Data       :List of 4
##   .. ..$ :List of 1
##   .. .. ..$ : chr [1:2858] "a lie" "abandon*" "abas*" "abattoir*" ...
##   .. ..$ :List of 1
##   .. .. ..$ : chr [1:1709] "ability*" "abound*" "absolv*" "absorbent*" ...
##   .. ..$ :List of 1
##   .. .. ..$ : chr [1:1721] "best not" "better not" "no damag*" "no no" ...
##   .. ..$ :List of 1
##   .. .. ..$ : chr [1:2860] "not a lie" "not abandon*" "not abas*" "not abattoir*" ...
```

```
##   ..@ concatenator: chr " "
```

```
#create a dfm of dictionary words found in the Cameron corpus
sentiment.dfm <- dfm(corpus.cameron.dfm,
                     dictionary = data_dictionary_LSD2015)
dim(sentiment.dfm)
```

```
## [1] 493   4
```

```
head(sentiment.dfm)
```

```
## Document-feature matrix of: 6 documents, 4 features (50.0% sparse).
## 6 x 4 sparse Matrix of class "dfm"
##        features
## docs    negative positive neg_positive neg_negative
##   text1       45      141            0            0
##   text2       10       49            0            0
##   text3        6       51            0            0
##   text4       63      250            0            0
##   text5       10       47            0            0
##   text6       32       18            0            0
```

We can calculate the proportion of negative sentiment words and positive sentiment words in each speech (*Question*: The columns that contain negations of positive sentiment and of negative sentiment contain zeroes. Why is this?), and we'll save those as variables in docvars.

```
#proportion of negative words
docvars(corpus.cameron.dfm, "prop.neg.words") <-
  as.numeric(sentiment.dfm[,1] / ntoken(corpus.cameron.dfm))

#proportion of positive words
docvars(corpus.cameron.dfm, "prop.pos.words") <-
  as.numeric(sentiment.dfm[,2] / ntoken(corpus.cameron.dfm))

#net sentiment
docvars(corpus.cameron.dfm, "net.sentiment")  <-
  docvars(corpus.cameron.dfm, "prop.pos.words") -
  docvars(corpus.cameron.dfm, "prop.neg.words")
```

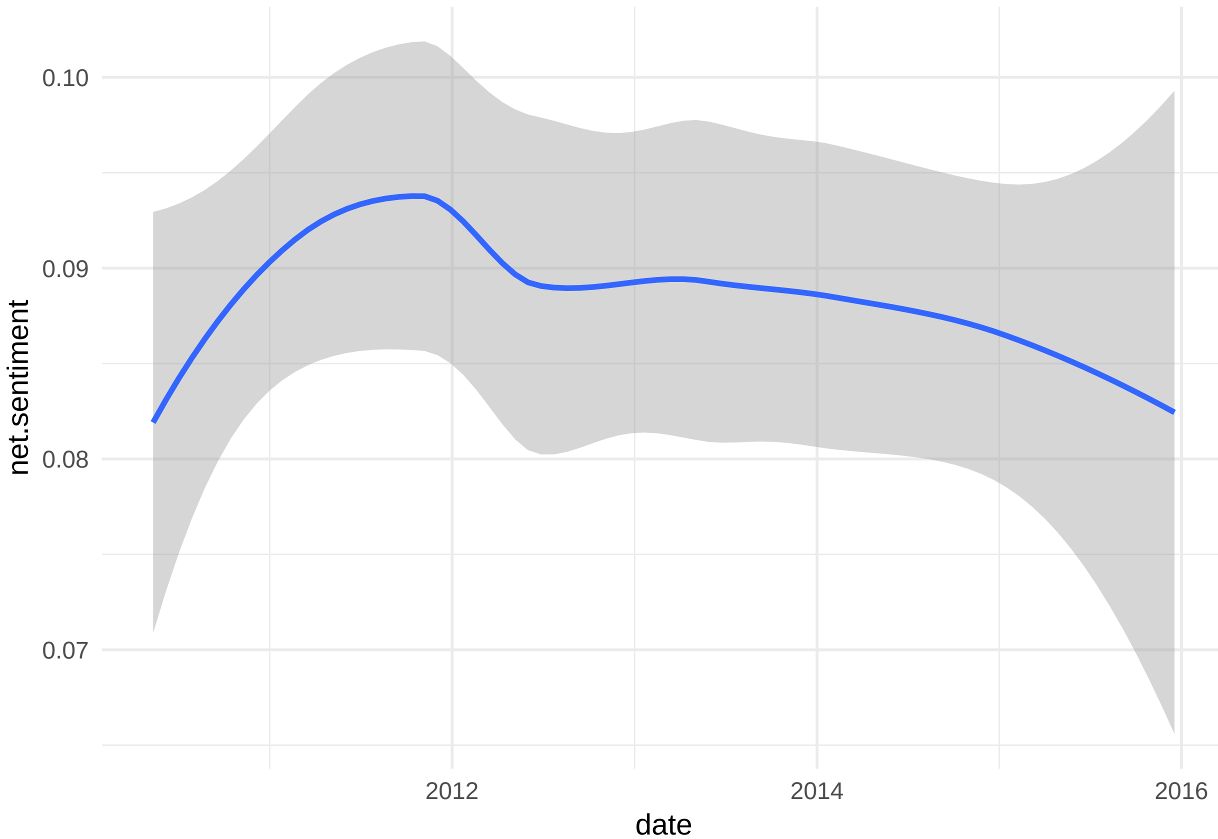
Did the net sentiment of Cameron speeches change over time? Let's plot to find out.

```
library(ggplot2)

sentiment.plot <- ggplot(docvars(corpus.cameron.dfm),
                         aes(x = date,
                             y = net.sentiment))

sentiment.plot  <- sentiment.plot  + geom_smooth() + theme_minimal()

print(sentiment.plot)
```



Answer: not really.

For more off-the-shelve dictionaries take a look at the `tidytext` library. `quanteda` can read those in as well.

Supervised machine learning

Perhaps there is something intrinsically different about how Cameron and Brown speak. Let's see if we build a classifier to predict if a speech is delivered by Cameron or Brown. First, take out a training set and a test from the corpus. The `set.seed()` function makes sure that you can replicate your random samples:

```
#set.seed() allows us to replicate randomly generated results
set.seed(2)

#generate random sample of 100 speeches of the corpus
corpus <- corpus_sample(corpus, 100, replace = FALSE)

#create id variable
docvars(corpus, "id_numeric") <- 1:ndoc(corpus)

#take note of how many speeches by Cameron and Brown
table(docvars(corpus, "speaker"))
```

```
##
## D. Cameron   G. Brown
##         65         35
```

Take a sample of 60 speeches as our training data and turn it into a stemmed dfm; the %>% operator is called a 'pipe': it takes the output of the preceding line of code as input to the following line of code.

```r
# generate 60 numbers without replacement
id_train <- sample(1:100, 60, replace = FALSE)
head(id_train, 10)
```

```
##  [1] 21 43 97 81 28 57 85 99 14 12
```

```r
train_dfm <- corpus_subset(corpus, id_numeric %in% id_train) %>%
    dfm(stem = TRUE)

#and the 40 remaining speeches as our test data.
test_dfm <- corpus_subset(corpus, !id_numeric %in% id_train) %>%
    dfm(stem = TRUE)

#check whether there is no overlap between the train set and the test set
which((docvars(train_dfm, "id_numeric")  %in% docvars(test_dfm, "id_numeric")))
```

```
## integer(0)
```

We can now train a Naive Bayes classifier on the training set:

```r
speaker.classifier <- textmodel_nb(train_dfm,
                                   y = docvars(train_dfm, "speaker"),
                                   smooth = 1,
                                   prior = "docfreq")

summary(speaker.classifier)
```

```
##
## Call:
## textmodel_nb.dfm(x = train_dfm, y = docvars(train_dfm, "speaker"),
##     smooth = 1, prior = "docfreq")
##
## Class Priors:
## (showing first 2 elements)
## D. Cameron   G. Brown
##       0.65       0.35
##
## Estimated Feature Scores:
##                 a transcript     of    the  prime minist podcast     on
## D. Cameron 0.6908     0.3014 0.6321 0.6292 0.4163 0.4472  0.1188 0.6616
## G. Brown   0.3092     0.6986 0.3679 0.3708 0.5837 0.5528  0.8812 0.3384
##            pre-budget report      , record     12 decemb  2009     .
## D. Cameron     0.2124 0.4688 0.6924 0.8119 0.5742 0.3929 0.102 0.715
## G. Brown       0.7876 0.5312 0.3076 0.1881 0.4258 0.6071 0.898 0.285
##              this   time last   year   mani  peopl were  worri     if
## D. Cameron 0.7074 0.5956 0.57 0.5665 0.6244 0.6371 0.58 0.4705 0.6867
## G. Brown   0.2926 0.4044 0.43 0.4335 0.3756 0.3629 0.42 0.5295 0.3133
##            they'd  still   have    job christma
## D. Cameron 0.4183 0.7036 0.5511 0.6254   0.3504
## G. Brown   0.5817 0.2964 0.4489 0.3746   0.6496
```

```r
head(coef(speaker.classifier))
```

```
##              classes
## features       D. Cameron  G. Brown
##   a             0.6907776 0.3092224
```

```
##   transcript  0.3014396 0.6985604
##   of          0.6321313 0.3678687
##   the         0.6291888 0.3708112
##   prime       0.4162621 0.5837379
##   minist      0.4472364 0.5527636
```

Let's analyze if we can predict whether a speech in the test set is from Cameron or Brown:

```
#Naive Bayes can only take features into consideration that occur both in the training set and the test

matched_dfm <- dfm_match(test_dfm, features = featnames(train_dfm))

#predict speaker
pred.speaker.classifier <- predict(speaker.classifier,
                                   newdata = matched_dfm,
                                   type = "class")

#predict probability
pred.prob.classifier <- predict(speaker.classifier,
                                newdata = matched_dfm,
                                type = "probability")

table(predicted.speaker = pred.speaker.classifier,
      actual.speaker = docvars(test_dfm, "speaker"))
```

```
##                    actual.speaker
## predicted.speaker D. Cameron G. Brown
##        D. Cameron         26        6
##        G. Brown            0        8
```

So it appears we are quite successful at predicting whether a speech is delivered by Cameron or Brown. Using the descriptive text statistics we discussed in class we can figure out which features distinguish between both speakers. For example, using the `tidyverse` library we can easily show some of the most distinctive words for both speakers (PcGW = "predicted class given word", or the probability that our classifier assigns a document to one speaker or another when observing that word)

```
library(tidyverse)

word_probs <- speaker.classifier$PcGw %>%
  as.matrix() %>%
  t() %>%
  as.data.frame() %>%
  mutate(feature = rownames(.))

names(word_probs)[1] <- "Cameron"
names(word_probs)[2] <- "Brown"

#distinctive Cameron words
word_probs %>%
  arrange(desc(Cameron)) %>%
  head(20)
```

```
##      Cameron      Brown   feature
## 1  0.9824859 0.01751405     syria
## 2  0.9719896 0.02801039         ...
## 3  0.9700273 0.02997273      isil
```

```
## 4  0.9690257 0.03097426    indian
## 5  0.9655748 0.03442518   cameron
## 6  0.9612585 0.03874146   charter
## 7  0.9612585 0.03874146 caribbean
## 8  0.9589723 0.04102774        eu
## 9  0.9577249 0.04227514   content
## 10 0.9577249 0.04227514     assad
## 11 0.9557047 0.04429527    syrian
## 12 0.9557047 0.04429527   tonight
## 13 0.9534818 0.04651815  manchest
## 14 0.9534818 0.04651815   languag
## 15 0.9534818 0.04651815      gavi
## 16 0.9510241 0.04897592 shouldn't
## 17 0.9510241 0.04897592   teacher
## 18 0.9510241 0.04897592   marriag
## 19 0.9482921 0.05170789     libya
## 20 0.9452373 0.05476266      wast
```

```r
#distinctive Brown words
word_probs %>%
  arrange(desc(Brown)) %>%
  head(20)
```

```
##       Cameron     Brown      feature
## 1  0.01238865 0.9876113       afghan
## 2  0.03471141 0.9652886       matern
## 3  0.04480245 0.9551976 minister:wel
## 4  0.05117887 0.9488211       karzai
## 5  0.05216885 0.9478311         nurs
## 6  0.05372777 0.9462722   minister:i
## 7  0.05372777 0.9462722       gordon
## 8  0.05967151 0.9403285        assur
## 9  0.06316541 0.9368346       rwanda
## 10 0.06709390 0.9329061   pittsburgh
## 11 0.06709390 0.9329061        brown
## 12 0.06709390 0.9329061       gilani
## 13 0.07801035 0.9219897        troop
## 14 0.08248383 0.9175162      helmand
## 15 0.08248383 0.9175162       berlin
## 16 0.08248383 0.9175162       fayyad
## 17 0.08931265 0.9106873      remuner
## 18 0.08931265 0.9106873       cancer
## 19 0.08931265 0.9106873      taleban
## 20 0.08931265 0.9106873      genocid
```