Session 3

Christian Kahmann

12 10 2020

This tutorial shows how to extract key terms from document and (sub-)collections with TF-IDF and the log-likelihood statistic and a reference corpus. We also show how it is possible to hande multi-word units such as 'United States' with the quanteda package.

1. Multi-word tokenization
2. TF-IDF
3. Log-likelihood ratio test
4. Visualization

Multi-word tokenization

Like in the previous tutorial we read the CSV data file containing the State of the union addresses and preprocess the corpus object with a sequence of `quanteda` functions.

In addition, we introduce handling of multi-word units (MWUs), also known as collocations in linguistics. MWUs are words comprising two or more semantically related tokens, such as `machine learning', which form a distinct new sense. Further, named entities such as`George Washington' can be regarded as collocations, too. They can be inferred automatically with a statistical test. If two terms occur significantly more often as direct neighbors as expected by chance, they can be treated as collocations.

Quanteda provides two functions for handling MWUs: `textstat_collocations` performs a statsictical test to identify collocation candidates. `tokens_compound` concatenates collocation terms in each document with a separation character, e.g. `_`. By this, the two terms are treated as a single new vocabulary type for any subsequent text processing algorithm.

Finally, we create a Document-Term-Matrix as usual, but this time with unigram tokens and concatenated MWU tokens.

```
options(stringsAsFactors = FALSE)
library(quanteda)

# read the SOTU corpus data
textdata <- read.csv("data/sotu.csv", sep = ";", encoding = "UTF-8")
sotu_corpus <- corpus(textdata$text, docnames = textdata$doc_id)

# Build a dictionary of lemmas
lemma_data <- read.csv("resources/baseform_en.tsv", encoding = "UTF-8")

# read an extended stop word list
stopwords_extended <- readLines("resources/stopwords_en.txt", encoding = "UTF-8")

# Preprocessing of the corpus
corpus_tokens <- sotu_corpus %>%
  tokens(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE) %>%
  tokens_tolower() %>%
  tokens_replace(lemma_data$inflected_form, lemma_data$lemma) %>%
  tokens_remove(pattern = stopwords_extended, padding = T)
```

```
# calculate multi-word unit candidates
sotu_collocations <- textstat_collocations(corpus_tokens, min_count = 25)
# check top collocations
head(sotu_collocations, 25)
```

```
##                collocation count count_nested length   lambda         z
## 1              unite state  4518            0      2 8.395507 157.29358
## 2              fiscal year   768            0      2 7.583265  78.51098
## 3           annual message   204            0      2 7.828220  77.37705
## 4                end june   223            0      2 6.937635  77.12023
## 5              health care   203            0      2 7.221388  76.93752
## 6       federal government   404            0      2 4.544370  75.98756
## 7              public debt   272            0      2 5.690518  75.06519
## 8          social security   196            0      2 7.087756  73.02594
## 9          american people   392            0      2 4.046357  72.38093
## 10               past year   304            0      2 4.943933  69.99174
## 11             public land   265            0      2 4.912068  69.91847
## 12                year end   315            0      2 4.637846  69.68812
## 13          billion dollar   156            0      2 7.294118  69.39964
## 14          million dollar   150            0      2 6.224792  63.62246
## 15                year ago   338            0      2 6.867191  61.38436
## 16            soviet union   124            0      2 7.119548  58.85244
## 17          fellow citizen   170            0      2 7.310087  58.34076
## 18             middle east   104            0      2 9.584624  56.72250
## 19         economic growth   105            0      2 6.275263  54.87274
## 20               arm force   123            0      2 5.694755  54.57533
## 21  commercial intercourse    90            0      2 6.784484  53.65897
## 22           supreme court   113            0      2 8.288329  53.64754
## 23     interstate commerce   107            0      2 7.542626  53.23642
## 24 favorable consideration    99            0      2 6.592423  53.19700
## 25         central america   107            0      2 6.574329  52.72316
```

```
# check bottom collocations
tail(sotu_collocations, 25)
```

```
##                collocation count count_nested length     lambda         z
## 471          good interest    34            0      2  1.9250729 11.178114
## 472         saddam hussein    27            0      2 16.5243323 11.165573
## 473           buenos ayres    31            0      2 16.2806403 11.137000
## 474           make america    34            0      2  1.9046181 11.033032
## 475               al qaeda    36            0      2 15.6588229 10.867428
## 476            state court    29            0      2  2.0357021 10.833873
## 477             rio grande    51            0      2 15.4825416 10.817689
## 478          santo domingo    29            0      2 15.3982762 10.675730
## 479       state government   104            0      2  1.0133097 10.227933
## 480       congress provide    30            0      2  1.8270888  9.972834
## 481              good work    30            0      2  1.8231888  9.965848
## 482      ballistic missile    25            0      2 14.0792658  9.824752
## 483     government program    29            0      2  1.6992497  9.103888
## 484             great work    31            0      2  1.6109863  8.955303
## 485       state department    36            0      2  1.4770656  8.809895
## 486             bering sea    26            0      2 12.3537479  8.648924
## 487          present state    45            0      2  1.2860206  8.575602
## 488 government expenditure    25            0      2  1.6993983  8.467038
## 489            great power    29            0      2  1.4807251  7.973107
```

```
## 490       present congress    26           0      2  1.3010836  6.645685
## 491        american nation    25           0      2  1.2501791  6.268707
## 492          foreign state    25           0      2  1.1771996  5.888551
## 493              make good    30           0      2  1.0402376  5.703403
## 494         american state    37           0      2  0.7563983  4.597798
## 495    american government    30           0      2  0.6792921  3.728092
```

Caution: For the calculation of collocation statistics being aware of deleted stop words, you need to add the paramter `padding = T` to the `tokens_remove` function above.

If you do not like all of the suggested collocation pairs to be considered as MWUs in the subsequent analysis, you can simply remove rows containing unwanted pairs from the sotu_collocations object.

```
# We will treat the top 250 collocations as MWU
sotu_collocations <- sotu_collocations[1:250, ]

# compound collocations
corpus_tokens <- tokens_compound(corpus_tokens, sotu_collocations)

# Create DTM (also remove padding empty term)
DTM <- corpus_tokens %>%
  tokens_remove("") %>%
  dfm()
```

TF-IDF

A widely used method to weight terms according to their semantic contribution to a document is the **term frequency–inverse document frequency** measure (TF-IDF). The idea is, the more a term occurs in a document, the more contributing it is. At the same time, in the more documents a term occurs, the less informative it is for a single document. The product of both measures is the resulting weight.

Let us compute TF-IDF weights for all terms in the first speech of Barack Obama.

```
# Compute IDF: log(N / n_i)
number_of_docs <- nrow(DTM)
term_in_docs <- colSums(DTM > 0)
idf <- log2(number_of_docs / term_in_docs)

# Compute TF
first_obama_speech <- which(textdata$president == "Barack Obama")[1]
tf <- as.vector(DTM[first_obama_speech, ])

# Compute TF-IDF
tf_idf <- tf * idf
names(tf_idf) <- colnames(DTM)
```

The last operation is to append the column names again to the resulting term weight vector. If we now sort the tf-idf weights decreasingly, we get the most important terms for the Obama speech, according to this weight.

```
sort(tf_idf, decreasing = T)[1:20]
```

```
## health_care    re-start         job        lend     tonight    recovery
##    39.54990    31.45674    28.29379    23.89260    23.80121    22.34426
##      layoff      ensure     college   renewable   recession      budget
##    20.59256    20.05307    19.81543    18.16903    16.16557    15.90033
```

```
##      crisis     inherit   long-term high_school accountable     quitter
##    15.81714    15.45674    15.03980    14.32093    13.88747    13.72837
##        auto        iraq
##    13.62677    13.61902
```

If we would have just relied upon term frequency, we would have obtained a list of stop words as most important terms. By re-weighting with inverse document frequency, we can see a heavy focus on business terms in the first speech. By the way, the quanteda-package provides a convenient function for computing tf-idf weights of a given DTM: `dfm_tfidf(DTM)`.

Log likelihood

We now use a more sophisticated method with a comparison corpus and the log likelihood statistic.

```
targetDTM <- DTM

termCountsTarget <- as.vector(targetDTM[first_obama_speech, ])
names(termCountsTarget) <- colnames(targetDTM)
# Just keep counts greater than zero
termCountsTarget <- termCountsTarget[termCountsTarget > 0]
```

In *termCountsTarget* we have the tf for the first Obama speech again.

As a comparison corpus, we select a corpus from the Leipzig Corpora Collection (http://corpora.uni-leipzig.de): 30.000 randomly selected sentences from the Wikipedia of 2010. **CAUTION:** The preprocessing of the comparison corpus must be identical to the preprocessing Of the target corpus to achieve meaningful results!

```
lines <- readLines("resources/eng_wikipedia_2010_30K-sentences.txt", encoding = "UTF-8")
corpus_compare <- corpus(lines)
```

From the comparison corpus, we also create a count of all terms.

```
# Create a DTM (may take a while)
corpus_compare_tokens <- corpus_compare %>%
  tokens(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE) %>%
  tokens_tolower() %>%
  tokens_replace(lemma_data$inflected_form, lemma_data$lemma) %>%
  tokens_remove(pattern = stopwords_extended, padding = T)

# Create DTM
comparisonDTM <- corpus_compare_tokens %>%
  tokens_compound(sotu_collocations) %>%
  tokens_remove("") %>%
  dfm()

termCountsComparison <- colSums(comparisonDTM)
```

In *termCountsComparison* we now have the frequencies of all (target) terms in the comparison corpus.

Let us now calculate the log-likelihood ratio test by comparing frequencies of a term in both corpora, taking the size of both corpora into account. First for a single term:

```
# Loglikelihood for a single term
term <- "health_care"

# Determine variables
a <- termCountsTarget[term]
```

```
b <- termCountsComparison[term]
c <- sum(termCountsTarget)
d <- sum(termCountsComparison)

# Compute log likelihood test
Expected1 = c * (a+b) / (c+d)
Expected2 = d * (a+b) / (c+d)
t1 <- a * log((a/Expected1))
t2 <- b * log((b/Expected2))
logLikelihood <- 2 * (t1 + t2)

print(logLikelihood)
```

```
## health_care
##    121.1461
```

The LL value indicates whether the term occurs significantly more frequently / less frequently in the target counts than we would expect from the observation in the comparative counts. Specific significance thresholds are defined for the LL values:

- 95th percentile; 5% level; $p < 0.05$; critical value = 3.84
- 99th percentile; 1% level; $p < 0.01$; critical value = 6.63
- 99.9th percentile; 0.1% level; $p < 0.001$; critical value = 10.83
- 99.99th percentile; 0.01% level; $p < 0.0001$; critical value = 15.13

With R it is easy to calculate the LL-value for all terms at once. This is possible because many computing operations in R can be applied not only to individual values, but to entire vectors and matrices. For example, `a / 2` results in a single value *a divided by 2* if `a` is a single number. If `a` is a vector, the result is also a vector, in which all values are divided by 2.

ATTENTION: A comparison of term occurrences between two documents/corpora is actually only useful if the term occurs in both units. Since, however, we also want to include terms which are not contained in the comparative corpus (the `termCountsComparison` vector contains 0 values for these terms), we simply add 1 to all counts during the test. This is necessary to avoid `NaN` values which otherwise would result from the log-function on 0-values during the LL test. Alternatively, the test could be performed only on terms that actually occur in both corpora.

First, let's have a look into the set of terms only occurring in the target document, but not in the comparison corpus.

```
# use set operation to get terms only occurring in target document
uniqueTerms <- setdiff(names(termCountsTarget), names(termCountsComparison))
# Have a look into a random selection of terms unique in the target corpus
sample(uniqueTerms, 20)
```

```
##  [1] "decency"           "market-based"       "short-cuts"
##  [4] "orrin"             "re-tooled"          "candidly"
##  [7] "god_bless"         "vigilant"           "biden"
## [10] "out-compete"       "flawlessly"         "inaction"
## [13] "speculator"        "private_enterprise" "sleepless"
## [16] "equivocation"      "predicament"        "re-finance"
## [19] "agribusiness"      "common-sense"
```

Now we calculate the statistics the same way as above, but with vectors. But, since there might be terms in the targetCounts which we did not observe in the comparison corpus, we need to make both vocabularies matching. For this, we append unique terms from the target as zero counts to the comparison frequency vector.

Moreover, we use a little trick to check for zero counts of frequency values in a or b when computing t1 or t2. If a count is zero the log function would produce an NaN value, which we want to avoid. In this case the `a == 0` resp. `b == 0` expression add 1 to the expression which yields a 0 value after applying the log function.

```
# Create vector of zeros to append to comparison counts
zeroCounts <- rep(0, length(uniqueTerms))
names(zeroCounts) <- uniqueTerms
termCountsComparison <- c(termCountsComparison, zeroCounts)

# Get list of terms to compare from intersection of target and comparison vocabulary
termsToCompare <- intersect(names(termCountsTarget), names(termCountsComparison))

# Calculate statistics (same as above, but now with vectors!)
a <- termCountsTarget[termsToCompare]
b <- termCountsComparison[termsToCompare]
c <- sum(termCountsTarget)
d <- sum(termCountsComparison)
Expected1 = c * (a+b) / (c+d)
Expected2 = d * (a+b) / (c+d)
t1 <- a * log((a/Expected1) + (a == 0))
t2 <- b * log((b/Expected2) + (b == 0))
logLikelihood <- 2 * (t1 + t2)

# Compare relative frequencies to indicate over/underuse
relA <- a / c
relB <- b / d
# underused terms are multiplied by -1
logLikelihood[relA < relB] <- logLikelihood[relA < relB] * -1
```

Let's take a look at the results: The 50 more frequently used / less frequently used terms, and then the more frequently used terms compared to their frequency. We also see terms that have comparatively low frequencies are identified by the LL test as statistically significant compared to the reference corpus.

```
# top terms (overuse in targetCorpus compared to comparisonCorpus)
sort(logLikelihood, decreasing=TRUE)[1:50]
```

```
##     health_care        american         economy             job
##       121.14613       110.85756       101.27230        87.63369
##         tonight         america          budget        recovery
##        85.01941        67.90865        67.60581        66.14296
##          crisis            lend         deficit            plan
##        65.33199        62.73237        57.98055        55.26975
##          reform            cost  responsibility          nation
##        54.97901        53.80704        53.07413        51.05914
##        congress          energy       education          afford
##        48.28926        45.79050        42.83649        42.39513
##       recession american_people      confidence            bank
##        41.82158        40.29198        40.07235        39.42911
##     accountable        re-start       long-term          invest
##        38.90939        38.90939        36.46019        34.86676
##            loan          ensure         tax_cut          dollar
##        34.38881        34.17352        33.92087        33.49738
##      prosperity            debt        medicare             bad
##        31.43753        30.58356        29.18204        28.97788
##         country          future        taxpayer       renewable
```

```
##        27.79142        25.61839        25.54218        25.54218
##           money             buy          layoff           spend
##        25.36427        24.92780        24.69887        23.04517
##         college        business        economic         inherit
##        22.27778        21.89767        20.63075        20.56073
##       financial      investment
##        20.19135        20.10559
```

```
# bottom terms (underuse in targetCorpus compared to comparisonCorpus)
sort(logLikelihood, decreasing=FALSE)[1:25]
```

```
##       game       city     follow      early        win       numb
## -3.7519628 -3.5736548 -2.5277954 -2.4613495 -1.8702817 -1.8079878
##      state      point      leave       show       book     record
## -1.6920659 -1.6543846 -1.5823090 -1.2507745 -1.1028102 -1.0663127
##       area    include university       type     design    control
## -1.0165030 -1.0073460 -0.8205736 -0.7869771 -0.7786415 -0.6410402
##        age        run      local      fight    produce    general
## -0.4614764 -0.4586085 -0.4403883 -0.4196276 -0.3992028 -0.3924707
##    attempt
## -0.3529017
```

```
llTop100 <- sort(logLikelihood, decreasing=TRUE)[1:100]
frqTop100 <- termCountsTarget[names(llTop100)]
frqLLcomparison <- data.frame(llTop100, frqTop100)
View(frqLLcomparison)

# Number of signficantly overused terms (p < 0.01)
sum(logLikelihood > 6.63)
```

```
## [1] 268
```

The method extracted 268 key terms from the first Obama speech.

Visualization

Finally, visualize the result of the 50 most significant terms as Wordcloud. This can be realized simply by function of the package wordcloud. Additionally to the words and their weights (here we use likelihood values), we override default scaling and color parameters. Feel free to try different parameters to modify the wordcloud rendering.

```
require(wordcloud2)
top50 <- sort(logLikelihood, decreasing = TRUE)[1:50]
top50_df <- data.frame(word = names(top50), count = top50, row.names = NULL)
wordcloud2(top50_df, shuffle = F, size = 0.5)
```

Alternative reference corpora

Key term extraction cannot be done for single documents, but for entire (sub-)corpora. Depending on the comparison corpora, the results may vary. Instead of comparing a single document to a Wikipedia corpus, we now compare collections of speeches of a single president, to speeches of all other presidents.

For this, we iterate over all different president names using a for-loop. Within the loop, we utilize a logical vector (Boolean TRUE/FALSE values), to split the DTM into two sub matrices: rows of the currently selected president and rows of all other presidents. From these matrices our counts of target and comparison frequencies are created. The statistical computation of the log-likelihood measure from above, we outsourced into the function `calculateLogLikelihood` which we load with the `source` command at the beginning of the block. The function just takes both frequency vectors as input parameters and outputs a LL-value for each term of the target vector.

Results of the LL key term extraction are visualized again as a wordcloud. Instead of plotting the wordcloud into RStudio, this time we write the visualization as a PDF-file to disk into the `wordclouds` folder. After the for-loop is completed, the folder should contain 42 wordcloud PDFs, one for each president.

```
source("resources/calculateLogLikelihood.R")

presidents <- unique(textdata$president)
```

```r
for (president in presidents) {

  cat("Extracting terms for president", president, "\n")

  selector_logical_idx <- textdata$president == president

  presidentDTM <- targetDTM[selector_logical_idx, ]
  termCountsTarget <- colSums(presidentDTM)

  otherDTM <- targetDTM[!selector_logical_idx, ]
  termCountsComparison <- colSums(otherDTM)

  loglik_terms <- calculateLogLikelihood(termCountsTarget, termCountsComparison)

  top50 <- sort(loglik_terms, decreasing = TRUE)[1:50]

  fileName <- paste0("wordclouds/", president, ".pdf")
  pdf(fileName, width = 9, height = 7)
  wordcloud::wordcloud(names(top50), top50, max.words = 50, scale = c(3, .9), colors = RColorBrewer::br
  dev.off()

}
```

Putting it all together

table (data.frame), which displays the top 25 terms of all speeches by frequency, tf-idf and log likelihood in columns.

```r
source("resources/calculateLogLikelihood.R")

frq <- sort(colSums(targetDTM), decreasing = T)[1:25]
tfidf <- sort(colSums(targetDTM) * log2(nrow(targetDTM) / colSums(targetDTM > 0)), decreasing = T)[1:25]
ll <- sort(calculateLogLikelihood(colSums(targetDTM), colSums(comparisonDTM)), decreasing = T)[1:25]

df <- data.frame(
  word.frq = names(frq),
  frq = frq,
  word.tfidf = names(tfidf),
  tfidf = tfidf,
  word.ll = names(ll),
  ll = ll,
  row.names = NULL
)
head(df, 10)
```

```
##       word.frq  frq word.tfidf     tfidf     word.ll        ll
## 1   government 6595    program 1452.3750    congress 3072.4017
## 2         make 5871    tonight 1235.4992  government 2717.7389
## 3     congress 5040        job 1107.9252 unite_state 2009.6103
## 4  unite_state 4518     mexico  980.8994      nation 1676.8002
## 5        state 4314    america  887.1629     country 1502.7445
## 6      country 4285  territory  794.5879         law 1060.1963
## 7         year 4132   economic  781.1532       peace  955.4410
```

```
## 8       people 3766       bank  774.2042        duty  913.9243
## 9        great 3555       cent  752.0329       great  909.7663
## 10      nation 3319    subject  740.0194    interest  893.0237
```

wordcloud which compares Obama's last speech with all his other speeches.

```
obama_speeches <- which(textdata$president == "Barack Obama")

last_speech_id <- length(obama_speeches)
termCountsTarget <- colSums(DTM[obama_speeches[last_speech_id], ])
termCountsTarget <- termCountsTarget[termCountsTarget > 0]
termCountsComparison <- colSums(DTM[obama_speeches[-last_speech_id], ])
loglik_terms <- calculateLogLikelihood(termCountsTarget, termCountsComparison, minSignificance = 3.84)

top50 <- sort(loglik_terms, decreasing = TRUE)[1:50]
wordcloud::wordcloud(names(top50), top50, max.words = 50, scale = c(3, .9), colors = RColorBrewer::brew
```