Text mining in practice - Bag of Words - Common qdap visuals

Digital Humanities DSC 105 Spring 2023

Marcus Birkenkrahe

March 24, 2023

README

- This lecture closely follows the DataCamp lesson "Text Mining with Bag-of-Words in R" by Ted Kwartler.
- Download and open the practice file `7_visuals_qdap_practice.org` from GitHub to code along.
- In this lecture & practice:
  1. frequent terms with `tm` visualized using `barplot`
  2. frequent terms with `qdap` visualized using `plot`

Getting, loading, cleaning the corpus

- Download `corpora.org` from GitHub and run it: bit.ly/corpora_org
- You may have to check that your R console points at the right directory (use `getwd()` to check and `setwd()` to change)
- Includes corpus creation, corpus cleaning and check printing
- Check that all corpora are there by listing user-defined objects:

```
ls()

[1] "clean_corp"    "clean_corpus"  "coffee_corpus" "coffee_df"
[5] "coffee_dtm"    "coffee_m"      "coffee_src"    "coffee_tdm"
[9] "coffee_vec"    "coffee_wfm"    "i"
```

Frequent terms with `qdap` - lock and load

- With this approach you lose some control over the preprocessing steps but it's faster with `qdap::freq_terms`.

```
library(qdap)
search()
ls('package:qdap')
```

```
 [1] ".GlobalEnv"              "package:qdap"
 [3] "package:RColorBrewer"    "package:qdapTools"
 [5] "package:qdapRegex"       "package:qdapDictionaries"
 [7] "package:tm"              "package:NLP"
 [9] "package:pROC"            "ESSR"
[11] "package:stats"           "package:graphics"
[13] "package:grDevices"       "package:utils"
[15] "package:datasets"        "package:stringr"
[17] "package:httr"            "package:methods"
[19] "Autoloads"               "package:base"
  [1] "%&%"                          "%>%"
  [3] "%bs%"                         "%ex%"
  [5] "%sw%"                         "add_incomplete"
  [7] "add_s"                        "adjacency_matrix"
  [9] "adjmat"                       "all_words"
 [11] "Animate"                      "apply_as_df"
 [13] "apply_as_tm"                  "as.Corpus"
 [15] "as.DocumentTermMatrix"        "as.dtm"
 [17] "as.tdm"                       "as.TermDocumentMatrix"
 [19] "as.wfm"                       "automated_readability_index"
 [21] "bag_o_words"                  "beg2char"
 [23] "blank2NA"                     "boolean_search"
 [25] "bracketX"                     "bracketXtract"
 [27] "breaker"                      "build_qdap_vignette"
 [29] "capitalizer"                  "char_table"
 [31] "char2end"                     "character_count"
 [33] "character_table"              "check_spelling"
 [35] "check_spelling_interactive"   "check_text"
 [37] "chunker"                      "clean"
 [39] "cm_2long"                     "cm_code.blank"
 [41] "cm_code.combine"              "cm_code.exclude"
 [43] "cm_code.overlap"              "cm_code.transform"
 [45] "cm_combine.dummy"             "cm_df.fill"
 [47] "cm_df.temp"                   "cm_df.transcript"
 [49] "cm_df2long"                   "cm_distance"
 [51] "cm_dummy2long"                "cm_long2dummy"
 [53] "cm_range.temp"                "cm_range2long"
 [55] "cm_time.temp"                 "cm_time2long"
 [57] "colcomb2class"                "coleman_liau"
 [59] "colpaste2df"                  "colSplit"
 [61] "colsplit2df"                  "combo_syllable_sum"
 [63] "comma_spacer"                 "common"
 [65] "condense"                     "correct"
 [67] "counts"                       "cumulative"
 [69] "DATA"                         "DATA.SPLIT"
 [71] "DATA2"                        "delete"
 [73] "dir_map"                      "discourse_map"
 [75] "dispersion_plot"              "Dissimilarity"
 [77] "dist_tab"                     "diversity"
 [79] "duplicates"                   "edge_apply"
 [81] "end_inc"                      "end_mark"
 [83] "end_mark_by"                  "env.syl"
 [85] "exclude"                      "Filter"
 [87] "flesch_kincaid"               "folder"
 [89] "formality"                    "freq_terms"
 [91] "fry"                          "gantt"
 [93] "gantt_plot"                   "gantt_rep"
 [95] "gantt_wrap"                   "genX"
 [97] "genXtract"                    "gradient_cloud"
 [99] "hamlet"                       "htruncdf"
[101] "imperative"                   "incomp"
[103] "incomplete_replace"           "inspect_text"
[105] "is.global"                    "key_merge"
[107] "kullback_leibler"             "lcolsplit2df"
[109] "left_just"                    "lexical_classification"
[111] "linsear_write"                "ltruncdf"
[113] "lview"                        "mcsv_r"
[115] "mcsv_w"                       "mgsub"
[117] "mraja1"                       "mraja1spl"
[119] "multigsub"                    "multiscale"
[121] "NAer"                         "name2sex"
[123] "Network"                      "new_project"
[125] "ngrams"                       "object_pronoun_type"
[127] "outlier_detect"               "outlier_labeler"
[129] "paste2"                       "phrase_net"
[131] "plot_gantt_base"              "polarity"
[133] "polysyllable_sum"             "pos"
[135] "pos_by"                       "pos_tags"
[137] "potential_NA"                 "preprocessed"
[139] "pres_debate_raw2012"          "pres_debates2012"
```

```
[141] "pronoun_type"             "prop"
[143] "proportions"              "qcombine"
[145] "qcv"                      "qdap_df"
[147] "qheat"                    "qprep"
[149] "qtheme"                   "question_type"
[151] "qview"                    "raj"
[153] "raj.act.1"                "raj.act.1POS"
[155] "raj.act.2"                "raj.act.3"
[157] "raj.act.4"                "raj.act.5"
[159] "raj.demographics"         "rajPOS"
[161] "rajSPLIT"                 "random_data"
[163] "random_sent"              "rank_freq_mplot"
[165] "rank_freq_plot"           "raw.time.span"
[167] "read.transcript"          "replace_abbreviation"
[169] "replace_contraction"      "replace_number"
[171] "replace_ordinal"          "replace_symbol"
[173] "replacer"                 "right_just"
[175] "rm_empty_row"             "rm_row"
[177] "rm_stop"                  "rm_stopwords"
[179] "sample.time.span"         "scores"
[181] "scrubber"                 "Search"
[183] "sent_detect"              "sent_detect_nlp"
[185] "sentCombine"              "sentiment_frame"
[187] "sentSplit"                "SMOG"
[189] "space_fill"               "spaste"
[191] "speakerSplit"             "stem_words"
[193] "stem2df"                  "stemmer"
[195] "strip"                    "strWrap"
[197] "sub_holder"               "subject_pronoun_type"
[199] "syllable_count"           "syllable_sum"
[201] "syn"                      "syn_frame"
[203] "synonyms"                 "synonyms_frame"
[205] "term_match"               "termco"
[207] "termco_c"                 "termco_d"
[209] "termco2mat"               "Text"
[211] "Text<-"                   "theme_badkitchen"
[213] "theme_cafe"               "theme_duskheat"
[215] "theme_grayscale"          "theme_greyscale"
[217] "theme_hipster"            "theme_nightheat"
[219] "theme_norah"              "Title"
```

```
[221] "Title<-"                      "TOT"
[223] "tot_plot"                     "trans_cloud"
[225] "trans_context"                "trans_venn"
[227] "Trim"                         "truncdf"
[229] "type_token_ratio"             "unbag"
[231] "unique_by"                    "vertex_apply"
[233] "visual"                       "wc"
[235] "weight"                       "wfdf"
[237] "wfm"                          "wfm_combine"
[239] "wfm_expanded"                 "which_misspelled"
[241] "word_associate"               "word_cor"
[243] "word_count"                   "word_diff_list"
[245] "word_length"                  "word_list"
[247] "word_network_plot"            "word_position"
[249] "word_proximity"               "word_split"
[251] "word_stats"
```

- Load `qdap` and check the arguments of `qdap::freq_terms`:

```
 ## load the qdap package
library(qdap)
 ## check out the arguments of freq_terms
args(freq_terms)

function (text.var, top = 20, at.least = 1, stopwords = NULL,
    extend = TRUE, ...)
NULL
```

- The arguments are not self-explanatory! Check out the help page for this function: do this in the R console, not in this file!

- From the help page: "finds the most frequently occurring terms in a text vector.":

  1. specify maximum terms to show with the `text.var` argument
  2. specify vector of stopwords to remove with `stopwords` argument
  3. specify minimum character length of included words with `at.least`

```
text.var
```

```
The text variable.

top
Top number of terms to show.

at.least
An integer indicating at least how many letters
a word must be to be included in the output.

stopwords
A character vector of words to remove from the text.
qdap has a number of data sets that can be used as stop words
including: Top200Words, Top100Words, Top25Words.
For the tm package's traditional English stop words use
tm::stopwords("english").
```

- Solutions:

```
library(qdap) ## load the qdap package
args(freq_terms) # check out help(freq_terms), too

function (text.var, top = 20, at.least = 1, stopwords = NULL,
    extend = TRUE, ...)
NULL
```

Extracting the frequency vector

- Create named frequency vector `f1` using `freq_terms` on the `text` vector from the `coffee_df` data frame:

  1. extract at most 10 words (`top`)
  2. words should have minimum length 3 (`at.least`)
  3. use the "Top200Words" stopwords dictionary. (`stopwords`)
  4. display the structure of vector `f1`.

```
## extract text with qdap::freq_terms
f1 <- freq_terms(text.var=coffee_df,
                 top = 10,
                 at.least = 3,
                 stopwords = "Top200Words")
## display structure of vector
str(f1)

Classes 'freq_terms', 'all_words' and 'data.frame': 10 obs. of  2 variables:
 $ WORD: chr  "false" "coffee" "for" "relnofollowtwitter" ...
 $ FREQ: num  2997 1004 781 600 381 ...
```

- Solutions:

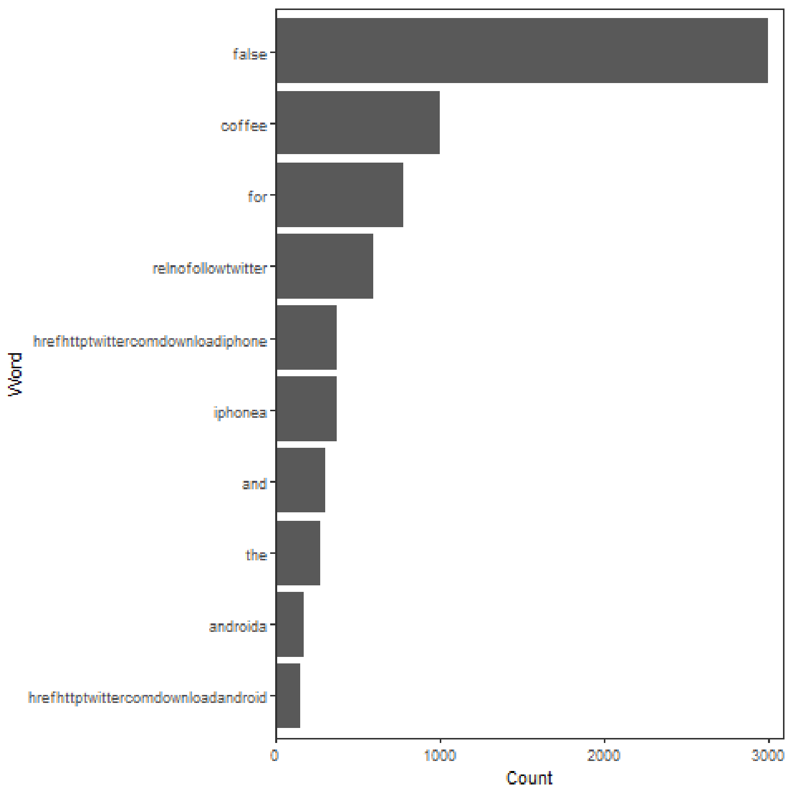
```
f1 <- freq_terms(text.var = coffee_df$text,
                 top = 10,
                 at.least = 3,
                 stopwords = "Top200Words")
str(f1)

Classes 'freq_terms', 'all_words' and 'data.frame': 10 obs. of  2 variables:
 $ WORD: chr  "coffee" "and" "the" "for" ...
 $ FREQ: num  1004 303 272 141 138 ...
```

Plotting with `plot`

- Making a basic plot of the results is easy. Just call `plot()` on the `freq_terms()` object. Because `plot` is generic, it will know that the frequency table should be plotted as a barchart.

- Produce a plot of `frequency` passing `f1` to `plot`:

```
plot(f1)
```

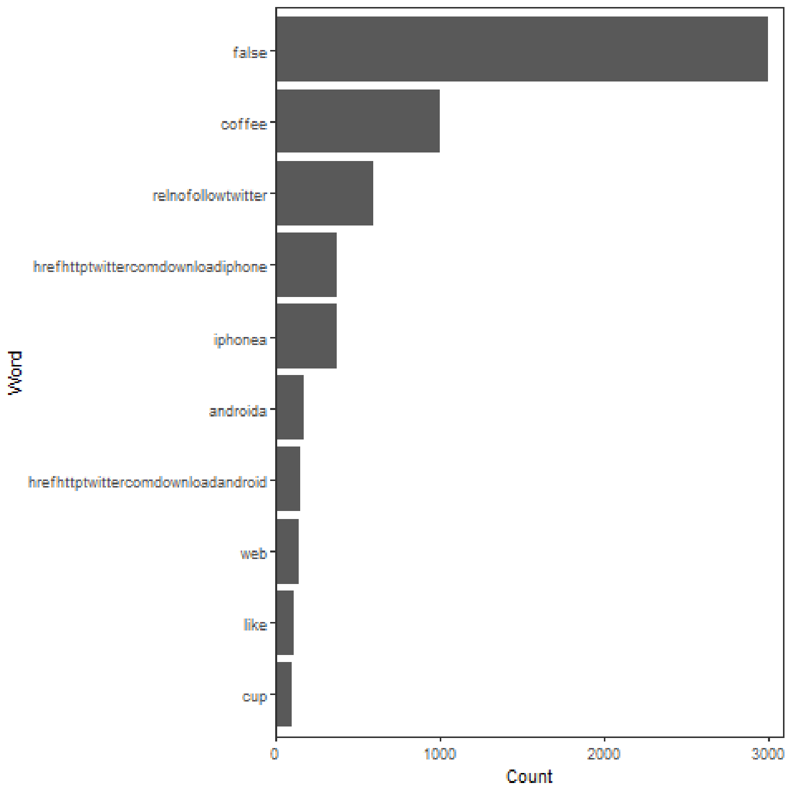


- Notice that there is no need to reorder the terms or tilt the plot. Unfortunately, the graph resists customization (title, etc.)
- Now produce another barplot with `plot`, but this time use the `stopwords("en")` dictionary. Create a vector `f2` with these properties and show the structure:

```
## define f2 as frequency vector with stopwords("en")
f2 <- freq_terms(text.var=coffee_df,
                 top = 10,
                 at.least = 3,
                 stopwords = stopwords("en"))
## display structure
str(f2)

Classes 'freq_terms', 'all_words' and 'data.frame': 10 obs. of  2 variables:
 $ WORD: chr  "false" "coffee" "relnofollowtwitter" "hrefhttptwittercomdownloadipl
 $ FREQ: num  2997 1004 600 381 381 ...
```

- Plot `f2` as before using `plot`:

  ```
  plot(f2)
  ```



- Look at the arguments:
  1. print `f1` and `f2`
  2. print the frequency `table` for both vectors
- Solutions:

  ```
  f2 <- freq_terms(text.var = coffee_df$text,
                   top = 10,
                   at.least = 3,
                   stopwords = stopwords("en"))
  str(f2)
  f1
  ```

```
f2
table(f1)
table(f2)

Classes 'freq_terms', 'all_words' and 'data.frame': 11 obs. of  2 variables:
 $ WORD: chr  "coffee" "like" "cup" "shop" ...
 $ FREQ: num  1004 111 103 69 66 ...
   WORD   FREQ
1  coffee 1004
2  and     303
3  the     272
4  for     141
5  you     138
6  like    111
7  have    107
8  cup     103
9  with    103
10 shop     69
   WORD     FREQ
1  coffee   1004
2  like      111
3  cup       103
4  shop       69
5  just       66
6  get        62
7  morning    57
8  want       49
9  drinking   47
10 can        45
11 looks      45
FREQ
WORD     69 103 107 111 138 141 272 303 1004
  and     0   0   0   0   0   0   0   1    0
  coffee  0   0   0   0   0   0   0   0    1
  cup     0   1   0   0   0   0   0   0    0
  for     0   0   0   0   0   1   0   0    0
  have    0   0   1   0   0   0   0   0    0
  like    0   0   0   1   0   0   0   0    0
  shop    1   0   0   0   0   0   0   0    0
  the     0   0   0   0   0   0   1   0    0
  with     0   1   0   0   0   0   0   0    0
  you      0   0   0   0   1   0   0   0    0
  FREQ
WORD        45 47 49 57 62 66 69 103 111 1004
  can        1  0  0  0  0  0  0   0   0    0
  coffee     0  0  0  0  0  0  0   0   0    1
  cup        0  0  0  0  0  0  0   1   0    0
  drinking   0  1  0  0  0  0  0   0   0    0
  get        0  0  0  0  1  0  0   0   0    0
  just       0  0  0  0  0  1  0   0   0    0
  like       0  0  0  0  0  0  0   0   1    0
  looks      1  0  0  0  0  0  0   0   0    0
  morning    0  0  0  1  0  0  0   0   0    0
  shop       0  0  0  0  0  0  1   0   0    0
  want       0  0  1  0  0  0  0   0   0    0
```