Text analysis with R and AmCAT

Wouter van Atteveldt

May 25, 2016

You might need these packages

```
install.packages("devtools")
devtools::install_github("amcat/amcat-r")
devtools::install_github("kasperwelbers/corpus-tools")
devtools::install_github("vanatteveldt/rsyntax")
```

Keyword queries with AmCAT

First, let's use the AmCAT API to do simple keyword queries:

On every computer you need to save your AmCAT password once (if you don't have an account you can create one for free at https://amcat.nl):

```
library(amcatr)
amcat.save.password("https://amcat.nl", "your_username", "your_password")
```

Next, you can connect using the `amcat.connect` function, storing the connection details in `conn`

```
conn = amcat.connect("https://amcat.nl")
```

You can use this connection to retrieve e.g. the meta-information about articles in a specific set:

```
meta = amcat.getarticlemeta(conn, 1006, 25173, dateparts = T)
head(meta)
```

```
##          id       date           medium       year      month       week
## 1 159408763 2016-02-17     De Telegraaf 2016-01-01 2016-02-01 2016-02-15
## 2 155888465 2015-11-19 Algemeen Dagblad 2015-01-01 2015-11-01 2015-11-16
## 3 157351072 2016-01-23    De Volkskrant 2016-01-01 2016-01-01 2016-01-18
## 4 157242055 2016-01-14 Algemeen Dagblad 2016-01-01 2016-01-01 2016-01-11
## 5 160812624 2016-03-09    De Volkskrant 2016-01-01 2016-03-01 2016-03-07
## 6 160812674 2016-03-09            Trouw 2016-01-01 2016-03-01 2016-03-07
```

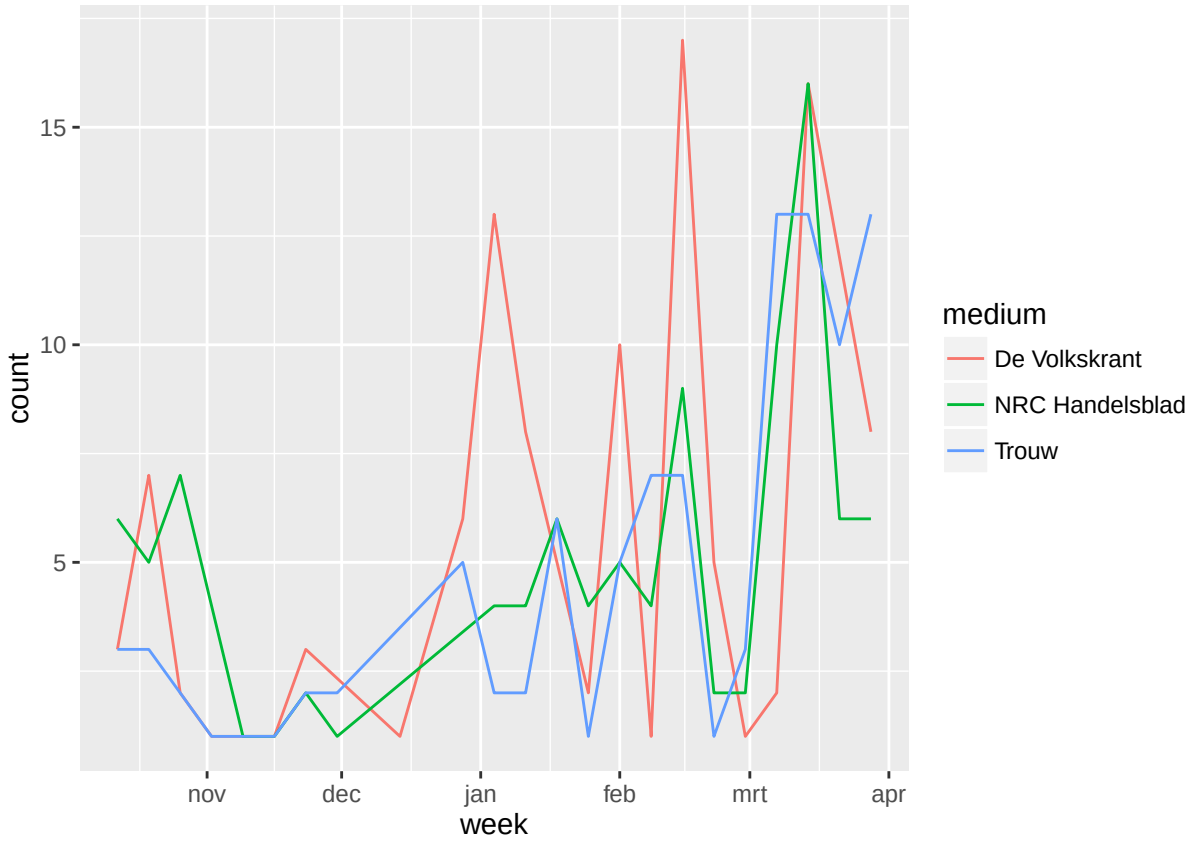
You can also use a keyword query which returns the number of hits per document for a query:

```
h = amcat.hits(conn, "referend*", sets=25173)
head(h)
```

```
##   count        id     query
## 1     1 155406567 referend*
## 2     1 155406700 referend*
## 3     1 155406925 referend*
## 4     1 155407049 referend*
## 5     1 155407220 referend*
## 6     1 155407243 referend*
```

Now, we can merge the information and plot the line over time:

```
meta = meta[meta$medium %in%  c("De Volkskrant", "Trouw", "NRC Handelsblad"),]
h = merge(meta, h)
perweek = aggregate(h["count"], h[c("week", "medium")], sum)
library(ggplot2)
ggplot(perweek, aes(x=week, y=count, color=medium)) + geom_line()
```



Corpus analysis: document-term matrix

The main primitive in corpus analysis is the document-term matrix. We can create one from text using the `create_matrix` function in `RTextTools`

```
library(RTextTools)
input = data.frame(text=c("Chickens are birds", "The bird eats"))
m = create_matrix(input$text, removeStopwords=F)
as.matrix(m)
```

```
##     Terms
## Docs are bird birds chickens eats the
##    1   1    0     1        1    0   0
##    2   0    1     0        0    1   1
```

Note that a DTM is normally a sparse matrix, which means only the non-zero values are stored. In a real world matrix, you easily have millions of cells, so converting it to a regular matrix with `as.matrix` can cause memory problems.

Let's try to clean some of the noise from the data set:

```
m = create_matrix(input$text, removeStopwords=T, stemWords=T, language='english')
as.matrix(m)
```

```
##     Terms
## Docs bird chicken eat
##    1    1       1   0
##    2    1       0   1
```

So for English this works reasonably well. Now let's try for Dutch:

```
text = c("De kip eet", "De kippen hebben gegeten")
m = create_matrix(text, removeStopwords=T, stemWords=T, language="dutch")
colSums(as.matrix(m))
```

```
##   eet geget   kip  kipp
##     1     1     1     1
```

Tokens and NLP Preprocessing

The `amcat.gettokens` command allows us to get document word lists from AmCAT:

```
tokens = amcat.gettokens(conn, 1006, 25173, page_size=100, max_page=2)
head(tokens)
```

```
##   position         term end_offset start_offset       aid
## 1        0        chaos          5            0 159408763
## 2        1         kiev         10            6 159408763
## 3        2     compleet         19           11 159408763
## 4        3         door         24           20 159408763
## 5        4       pieter         32           26 159408763
## 6        5 waterdrinker         45           33 159408763
```

We can create a DTM (and a word cloud) from this:

```
library(corpustools)
dtm = dtm.create(tokens$aid, tokens$term)
dtm.wordcloud(dtm)
```

But that's still no good: we need some preprocessing:

```
tokens = amcat.gettokens(conn, 1006, 25173, module="morphosyntactic", page_size=100, max_page=1, only_ca
head(tokens)
```

```
##           word parent para sentence term_id offset  pos        lemma
## 1        Chaos   <NA>    1        1     t_0      0 noun        chaos
## 2         Kiev    t_0    1        1     t_1      6 name         Kiev
## 3     compleet    t_0    1        1     t_2     11  adj     compleet
## 4         door    t_2    1        1     t_3     21 prep         door
## 5       Pieter    t_3    1        1     t_4     27 name       Pieter
## 6 Waterdrinker    t_4    1        1     t_5     34 name Waterdrinker
##   relation token_id       aid
## 1     <NA>       w1 159408763
## 2       dp       w2 159408763
## 3       dp       w3 159408763
## 4      mod       w4 159408763
## 5     obj1       w5 159408763
## 6      mwp       w6 159408763
```

So you can see this lemmatizes (stems) words and gives their part of speech (noun, verb, etc.) Let's plot only the names:

```
subset = tokens[tokens$pos == "name", ]
dtm = dtm.create(subset$aid, subset$lemma)
dtm.wordcloud(dtm, nterms = 200)
```

Corpus analysis: The State-of-the-Unions

We've prepared a data set containing state of the union speeches by Obama and Bush:

```
data(sotu)
head(sotu.tokens)
```

```
##         word sentence  pos      lemma offset       aid id pos1 freq
## 1         It        1  PRP         it      0 111541965  1    O    1
## 2         is        1  VBZ         be      3 111541965  2    V    1
## 3        our        1 PRP$         we      6 111541965  3    O    1
## 4 unfinished        1   JJ unfinished     10 111541965  4    A    1
## 5       task        1   NN       task     21 111541965  5    N    1
## 6         to        1   TO         to     26 111541965  6    ?    1
```

```
aggregate(cbind(Freq=sotu.meta$id), list(Speaker=sotu.meta$headline), length)
```

```
##          Speaker Freq
## 1   Barack Obama  554
## 2 George W. Bush  536
```

We can easily get the most frequent terms with the `term.statistics` function:

```
dtm = dtm.create(sotu.tokens$aid, sotu.tokens$lemma)
stats = term.statistics(dtm)
stats = arrange(stats, -termfreq)
head(stats)
```

```
##   term characters number nonalpha termfreq docfreq reldocfreq       tfidf
## 1  the          3  FALSE    FALSE     3847    1016  0.9321101 0.007516514
## 2  and          3  FALSE    FALSE     3261    1000  0.9174312 0.007949253
## 3 that          4  FALSE    FALSE     1376     712  0.6532110 0.022628960
## 4 have          4  FALSE    FALSE     1055     610  0.5596330 0.028151444
## 5 they          4  FALSE    FALSE      965     487  0.4467890 0.041932831
## 6  for          3  FALSE    FALSE      797     511  0.4688073 0.032917098
```

Let's limit that to adjectives:

```
dtm = with(subset(sotu.tokens, pos1 == "A"), dtm.create(aid, lemma))
stats = term.statistics(dtm)
stats = arrange(stats, -termfreq)
head(stats)
```

```
##       term characters number nonalpha termfreq docfreq reldocfreq
## 1      new          3  FALSE    FALSE      259     206 0.19845857
## 2     more          4  FALSE    FALSE      255     198 0.19075145
## 3 american          8  FALSE    FALSE      216     189 0.18208092
## 4     last          4  FALSE    FALSE      115     105 0.10115607
## 5     many          4  FALSE    FALSE      110      96 0.09248555
## 6     good          4  FALSE    FALSE      105      94 0.09055877
##       tfidf
## 1 0.5897179
## 2 0.5740020
## 3 0.6903693
## 4 0.7375330
## 5 0.9190045
## 6 0.8278900
```

Comparing corpora

It is often more informative to compare two corpora, e.g. compare Bush' words to Obama's words:

```
dtm = with(sotu.tokens[sotu.tokens$pos1 %in% c("N", "A", "M"), ],
           dtm.create(aid, lemma))
```

```
## Ignoring words with frequency lower than 5

## Ignoring words with less than 3 characters

## Ignoring words that contain numbers of non-word characters
```

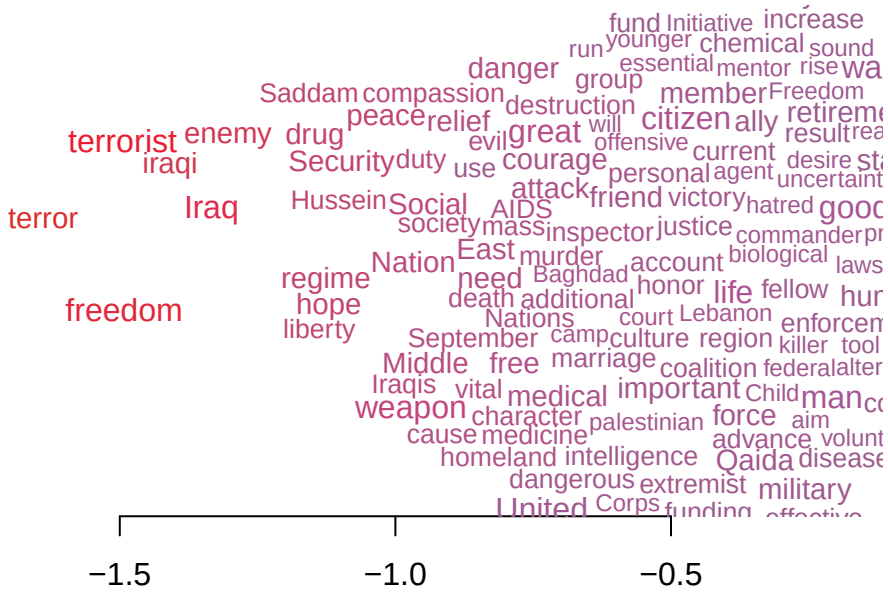
```
obama = sotu.meta$id[sotu.meta$headline == "Barack Obama"]
cmp  = corpora.compare(dtm, select.rows = obama)
cmp = arrange(cmp, over)
head(cmp)
```

```
##        term termfreq.x termfreq.y termfreq    relfreq.x   relfreq.y
## 1    terror          1         55       56 8.931761e-05 0.004610613
## 2 terrorist         13        103      116 1.161129e-03 0.008634420
## 3   freedom          8         79       87 7.145409e-04 0.006622517
## 4     iraqi          3         49       52 2.679528e-04 0.004107637
## 5     enemy          4         52       56 3.572705e-04 0.004359125
## 6      Iraq         15         94      109 1.339764e-03 0.007879956
##        over      chi
## 1 0.1941531 48.87172
## 2 0.2243133 64.62741
## 3 0.2249311 53.78511
## 4 0.2482465 37.95179
## 5 0.2532635 38.28727
## 6 0.2634883 52.65909
```

So, words like terror and freedom are mostly used by Bush (their overrepresentation for Obama is below 1).

We can also plot these words with a 'directed' word cloud:

```
h = rescale(log(cmp$over), c(1, .6666))
s = rescale(sqrt(cmp$chi), c(.25,1))
cmp$col = hsv(h, s, .33 + .67*s)
with(head(cmp, 130), plotWords(x=log(over), words=term,
    wordfreq=termfreq, random.y = T, col=col, scale=1))
```



Topic Modeling

A final example of corpus analysis is topic modeling. In topic modeling, words are automatically assigned to clusters (similar to factor analysis):

```
set.seed(123)
m = lda.fit(dtm, K=10, alpha=.1)
terms(m, 10)
```

```
##       Topic 1       Topic 2          Topic 3      Topic 4     Topic 5
##  [1,] "Iraq"        "people"         "energy"     "school"    "tax"
##  [2,] "man"         "time"           "year"       "child"     "year"
##  [3,] "country"     "american"       "new"        "education" "more"
##  [4,] "woman"       "nation"         "clean"      "college"   "family"
##  [5,] "year"        "America"        "research"   "student"   "job"
##  [6,] "Afghanistan" "country"        "oil"        "life"      "economy"
##  [7,] "military"    "Americans"      "technology" "year"      "business"
##  [8,] "war"         "other"          "more"       "America"   "american"
##  [9,] "troops"      "responsibility" "power"      "high"      "cut"
## [10,] "iraqi"       "same"           "America"    "community" "last"
##       Topic 6      Topic 7     Topic 8     Topic 9    Topic 10
##  [1,] "Congress"   "people"    "terrorist" "job"      "health"
##  [2,] "Security"   "America"   "weapon"    "new"      "care"
##  [3,] "Social"     "world"     "America"   "business" "budget"
##  [4,] "people"     "freedom"   "United"    "America"  "year"
##  [5,] "party"      "nation"    "nuclear"   "economy"  "next"
##  [6,] "reform"     "peace"     "world"     "world"    "insurance"
##  [7,] "law"        "great"     "States"    "american" "cost"
##  [8,] "member"     "free"      "regime"    "more"     "Congress"
##  [9,] "retirement" "States"    "country"   "place"    "Medicare"
## [10,] "american"   "democracy" "Iran"      "worker"   "plan"
```

We can see how often each topic occurs in each document:

```
tpd = topics.per.document(m, as.wordassignments = T)
head(tpd)
```

```
##          id X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## 1 111541965  0  6  0  0  0  1  0  0  0   0
## 2 111541995  0  3  0 10  3  0  0  2  7   0
## 3 111542001  1  1  0 19  6  0  1  0  0   1
## 4 111542006  0  0  0  9  0  4  2  0  0   2
## 5 111542013  1  0  0  9  0  4  2  0  7   1
## 6 111542018  0  0  0 10  1  0  0  0 12   0
```

And merge this back with the meta information

```
tpd = merge(sotu.meta, tpd)
head(tpd)
```

```
##          id   medium     headline       date X1 X2 X3 X4 X5 X6 X7 X8 X9
## 1 111541965 Speeches Barack Obama 2013-02-12  0  6  0  0  0  1  0  0  0
## 2 111541995 Speeches Barack Obama 2013-02-12  0  3  0 10  3  0  0  2  7
## 3 111542001 Speeches Barack Obama 2013-02-12  1  1  0 19  6  0  1  0  0
## 4 111542006 Speeches Barack Obama 2013-02-12  0  0  0  9  0  4  2  0  0
## 5 111542013 Speeches Barack Obama 2013-02-12  1  0  0  9  0  4  2  0  7
```

```
## 6 111542018 Speeches Barack Obama 2013-02-12  0  0  0 10  1  0  0  0 12
##   X10
## 1   0
## 2   0
## 3   1
## 4   2
## 5   1
## 6   0
```

And use this to e.g. figure out whether a topic like Iraq (topic 1) is mostly affiliated with which president:

```
t.test(tpd$X1 ~ tpd$headline)
```

```
##
##  Welch Two Sample t-test
##
## data:  tpd$X1 by tpd$headline
## t = -5.6213, df = 921.72, p-value = 2.512e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.8617571 -0.8981842
## sample estimates:
##   mean in group Barack Obama mean in group George W. Bush
##                     1.148014                     2.527985
```