Using Corpustools

Corpustools is an R package developed at the VU for analysing corpora while keeping word order. This allows functions such as keyword-in-context and windowed (proximity) search and networks that are impossible using DTM's.

Installing Corpustools

```
install.packages(c("wordcloud", "topicmodels", "LDAvis", "slam"))
devtools::install_github("kasperwelbers/corpustools")
```

```
library(corpustools)
```

The main class used by corpustools is tCorpus, representing a tokenized corpus. To get help on corpustools, use:

```
?tCorpus
```

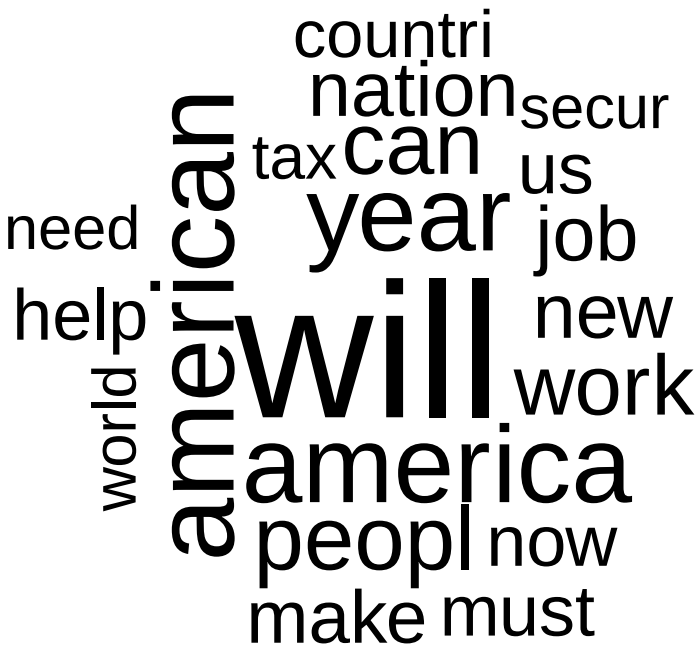
Creating and cleaning a tCorpus

The function `create_tcorpus` creates a tCorpus from a character vector (or data frame). As an example througout this document, we will use the `sotu_texts` objects from the corpustools package, which contains the State of the Union speeches by Obama and Bush.

The following code creates a tcorpus, adds a 'stem' feature containing the stemmed words with stopwords removed, converts it to a regular dtm and makes a word cloud:

```
tc = create_tcorpus(sotu_texts)
tc$preprocess('token', 'stem',
    remove_stopwords = TRUE, use_stemming = TRUE)

dtm = tc$dtm(feature="stem")
dtm_wordcloud(dtm, nterms = 20)
```



Dictionary search

`tCorpus` has a powerful boolean query language which also allows proximity search using lucene notation. For example, the following code finds all mentions of `terror*` and `america*` within 10 words of each other:

\begin{verbatim}

```
hits = tc$search_features('"terror* america*"~10')
head(hits$hits)
```

code	feature	doc_id	sent_i	token_i	hit_id
query_1	terror	1024	NA	27	1
query_1	American	1024	NA	36	1
query_1	terrorist	105	NA	34	2
query_1	American	105	NA	40	2
query_1	American	108	NA	2	3
query_1	Terrorists	108	NA	12	3

You can also automatically find words that occur together with a query or hits object:

```
words = tc$feature_associations(query="terror*")
# or: tc$feature_associations(hits=hits)
head(words)
```

	feature	freq	freq_NOT	ratio	chi2
1112	terrorists	81	0	13024.438055	1488.3126
1108	terror	56	0	9009.506472	1028.6735
1110	terrorist	33	0	5315.769416	606.0290
1109	terrorism	10	0	1622.032359	164.7490
1214	war	25	54	7.451001	113.1680
921	regimes	10	4	39.561765	112.3315

Context features

Since tCorpus keeps word order information, we can use this to extract contextual information. For example, the following gives the 'keyword-in-context' list of a query:

```
tc$kwic(query="terror*", n = 10)
```

doc_id	code	hit_id	feature	kwic
101		1	terror	... thanks to them, we are winning the war on . The men and women of our Armed F
105		103	terrorists	... Our soldiers, working with the Bosnian Government, seized who were plotting to b
102		2	terror	... are acting forcefully. Pakistan is now cracking down on , and I admire the strong l
102		3	terror	... But some governments will be timid in the face of . And make no mistake about it
1024		4	terror	... Syrian people deserve, a future free of dictatorship, , and fear. As we speak, Ameri
1011		68	terrorist	... succeed. We are clear eyed about Iran's support for organizations like Hizballah, w
1012		69	terrorist	..., we'll support the opposition that rejects the agenda of networks. Here at home, w
102		70	terrorist	... that all nations will heed our call and eliminate the parasites who threaten their co
1042		71	terrorist	... So even as we actively and aggressively pursue networks through more targeted eff
105		72	terrorist	... helping to train that country's armed forces to go after cells that have executed an

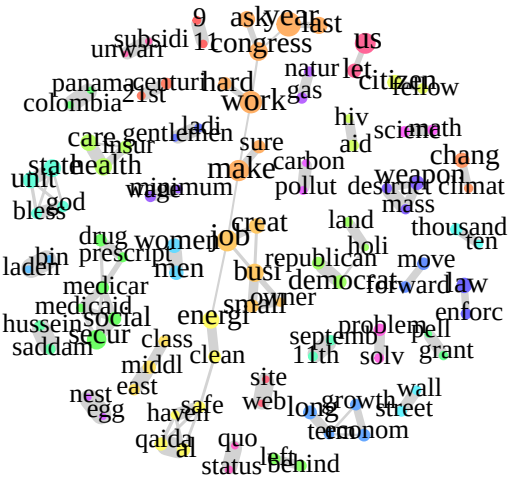
You can also filter a tcorpus on words occuring close to a query, for example to create a DTM of all terms close to a topic or actor of interest:

```
tc_subset = tc$subset_query(query = 'terror*', copy = T, window = 20)
dtm = tc_subset$dtm(feature="stem")
dtm_wordcloud(dtm, nterms = 20)
```

Semantic networks

You can also easily create a semantic network, i.e. the network of all terms that occur in proximity of each other:

```
g = tc$semnet_window('stem', window.size = 10)
g = backbone_filter(g, max_vertices = 100)
plot_semnet(g, return_graph = F)
```



See `?plot_semnet` for a list of all plotting and clustering options.

Corpus Analysis

Finally, you can also use `corpustools` to do 'normal' corpus analysis, such as comparing corpora and doing topic modeling (word clustering):

Comparing Corpora

For example, to compare Obama's words to Bush' we can use:

```
comp = tc$compare_subset('stem',
  subset_meta_x = president == 'Barack Obama')
comp = plyr::arrange(comp, -ratio)
head(comp)
```

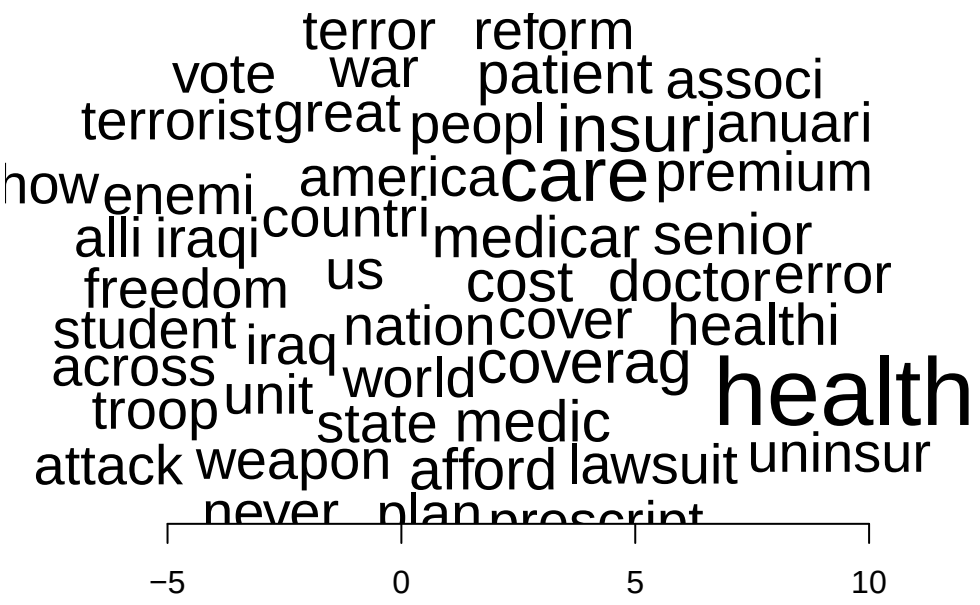
feature	freq.x	freq.y	freq	p.x	p.y	ratio	chi2
partnership	15	0	15	0.0006843	4.4e-06	154.4284	15.35240
graduat	13	0	13	0.0005937	4.4e-06	133.9743	13.30481
lend	13	0	13	0.0005937	4.4e-06	133.9743	13.30481
town	13	0	13	0.0005937	4.4e-06	133.9743	13.30481
tech	12	0	12	0.0005484	4.4e-06	123.7473	12.28108
forg	11	0	11	0.0005030	4.4e-06	113.5202	11.25740

You can also compare documents matching a query to those that don't:

```
comp = tc$compare_subset('stem',  query_x = "health*")
```

And plot the result as a comparison plot, showing the distinguishing words from the matching documents on the right and the reverse on the left:

```
plot(comp, n=50)
```



Topic Models

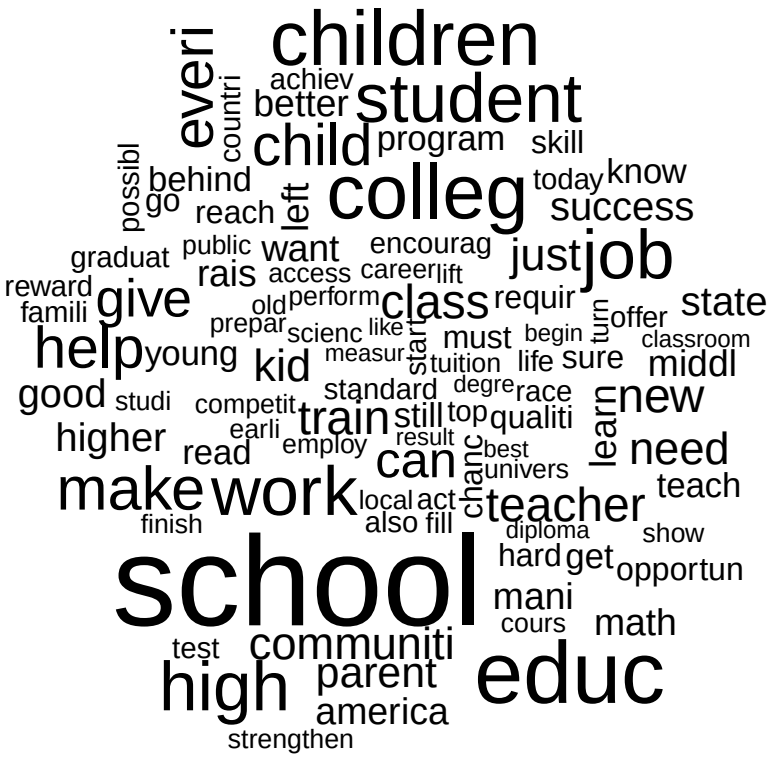
Finally, we can fit a topic model directly from a tcorpus object:

```
library(topicmodels)
m = tc$lda_fit('stem', K = 10, alpha = .1)
terms(m, 10)
```

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
help	new	america	health	will	year	one	terrorist	school	will
ask	job	nation	will	now	tax	american	weapon	educ	iraq
new	energi	us	care	make	american	women	unit	colleg	peopl
govern	america	must	must	can	will	men	world	children	iraqi
peopl	will	peopl	secur	get	million	day	america	job	forc
work	busi	will	budget	us	pay	year	secur	high	freedom
need	creat	world	reform	peopl	cut	us	threat	student	peac
america	year	can	need	right	famili	home	nuclear	work	support
tonight	american	great	congress	know	job	live	attack	make	secur
aid	invest	american	can	work	time	countri	nation	child	afghanistan

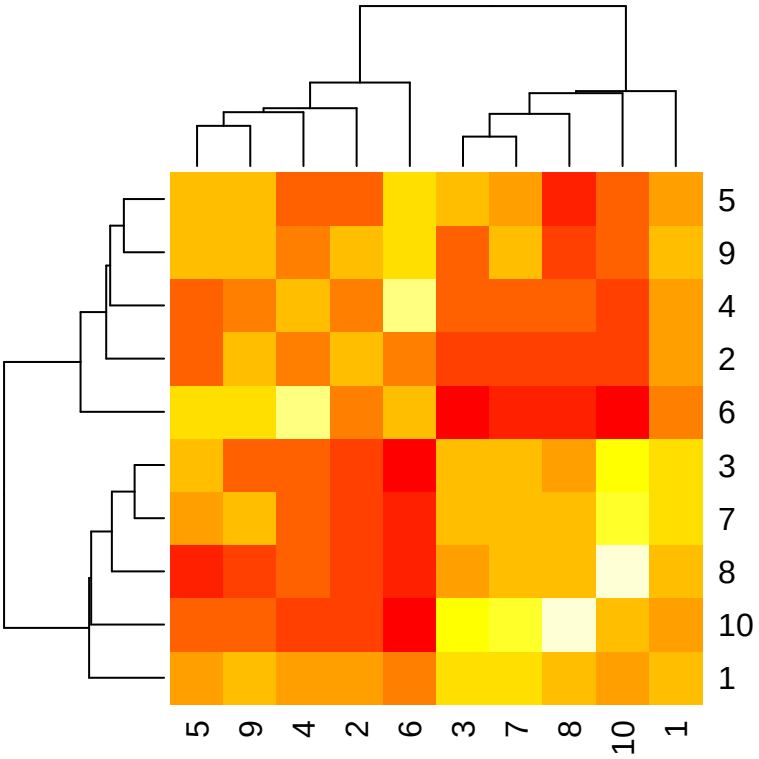
To extract words per topic, e.g. to make a word cloud for a topic, use the posterior function

```
library(wordcloud)
x = posterior(m)$terms[9, ]
x = sort(x, decreasing = T)[1:100]
x = x[!is.na(x)]
wordcloud(names(x), x)
```



To extract the topics per documents, for example top see topic correlations, use the following:

```
library(reshape2)
assignments = data.frame(i=m@wordassignments$i, j=m@wordassignments$j, v=m@wordassignments$v)
docsums = acast(assignments, i ~ v, value.var='j', fun.aggregate=length)
correlations = cor(docsums)
diag(correlations) = 0
heatmap(correlations, symm=T)
```



Finally, LDAvis has a nice interactive visualization of topic models. Unfortunately, this can't be done directly from tcorpus.

```
dtm = tc$dtm('stem')
m = LDA(dtm, k=10)

dtm = dtm[slam::row_sums(dtm) > 0, ]
phi = as.matrix(posterior(m)$terms)
theta <- as.matrix(posterior(m)$topics)
vocab <- colnames(phi)
doc.length = slam::row_sums(dtm)
term.freq = slam::col_sums(dtm)[match(vocab, colnames(dtm))]

library(LDAvis)
json = createJSON(phi = phi, theta = theta,
                  vocab = vocab,
                  doc.length = doc.length,
                  term.frequency = term.freq)
serVis(json)
```