Semantic Network Analysis

Kasper Welbers & Wouter van Atteveldt

May 25, 2016

One way to create semantic networks is to calculate how often words co-occur together. This co-occurence reflects a semantic relation, because it indicates that the meaning of these words is related.

In this howto we demonstrate two functions to calculate the co-occurence of words. The first is the `coOccurenceNetwork` function, which calculates the co-occurence of words within documents based on a document term matrix. The second is the `windowedCoOccurenceNetwork`, which calculates how often words co-occur within a given word distance based on tokenized texts.

We start with a simple example.

```
library(semnet)
data(simple_dtm)
dtm
```

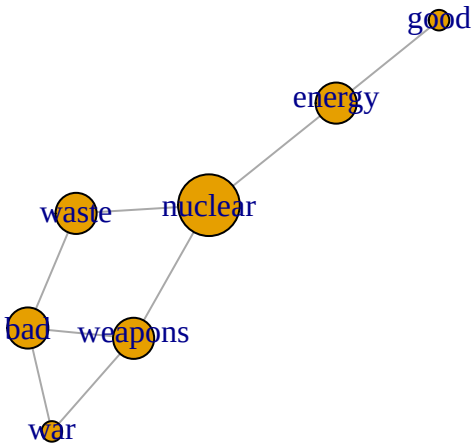
```
## 6 x 7 sparse Matrix of class "dgTMatrix"
##   nuclear energy waste weapons bad war good
## 1       1      1     .       .   .   .    .
## 2       1      .     1       .   .   .    .
## 3       1      .     .       1   .   .    .
## 4       .      .     .       1   1   1    .
## 5       .      .     1       .   1   .    .
## 6       .      1     .       .   .   .    1
```

`dtm` is a document term matrix: the rows represent documents and the columns represent words. Values represent how often a word occured within a document. The co-occurence of words can then be calculated as the number of documents in which two words occur together. This is what the `coOccurenceNetwork` function does.

```
g = coOccurenceNetwork(dtm)
```

```
## Note: method with signature 'CsparseMatrix#Matrix#missing#replValue' chosen for function '[<-',
##  target signature 'dgCMatrix#nsCMatrix#missing#numeric'.
##  "Matrix#nsparseMatrix#missing#replValue" would also be valid
```

```
plot(g, vertex.size=V(g)$freq*10)
```



Of course, this method mainly becomes interesting when lots of documents are analyzed. This could for instance show how often the word 'nuclear' is used in the context of 'energy', compared to the context of 'weapons' and 'waste'. Thus, it can provide an answer to the question: if one think or talks about nuclear technology, what discourses, frames or topics come to mind?

To demonstrate the `windowedCoOccurenceNetwork` function we'll use a larger dataset, consisting of the state of the union speeches of Obama and Bush (1090 paragraphs). We'll filter the data on part-of-speech tags to contain only the nouns, names and adjectives.

```
data(sotu)
sotu.tokens = sotu.tokens[sotu.tokens$pos1 %in% c('N','M','A'),]
head(sotu.tokens)
```

	word	sentence	pos	lemma	offset	aid	id	pos1	freq
4	unfinished	1	JJ	unfinished	10	111541965	4	A	1
5	task	1	NN	task	21	111541965	5	N	1
9	basic	1	JJ	basic	41	111541965	9	A	1
10	bargain	1	NN	bargain	47	111541965	10	N	1
14	country	1	NN	country	71	111541965	14	N	1
17	idea	1	NN	idea	84	111541965	17	N	1

We are interested in three columns in the `sotu.tokens` dataframe: * The `lemma` column, which is the lemma of a term (the non-plural basic form of a word). We use this instead of the word because we are interested in the meaning of words, for which it is generally less relevant in what specific form it is used. Thus, we consider the words "responsibility" and "responsibilities" to represent the same meaning. * The `aid` column, which is a unique id for the document, in this case for a paragraph in the SotU speeches. We refer to this as the `context` in which a word occurs. * The `id` column, which is the specific location of a term within a context. For example, the first row in sotu.tokens shows that in context `111541965`, the term `unfinished` was the fourth term.

These columns are the main arguments for the `windowedCoOccurenceNetwork` function. In addition, the `window.size` argument determines the word distance within which words need to occur to be counted as a co-occurence.

```
g = windowedCoOccurenceNetwork(location=sotu.tokens$id,
                      term=sotu.tokens$lemma,
                      context=sotu.tokens$aid,
                      window.size=20)
```

```
class(g)
```

```
## [1] "igraph"
```

```
vcount(g)
```

```
## [1] 3976
```

```
ecount(g)
```

```
## [1] 201792
```

Visualizing Semantic Networks

The output `g` is an igraph object—a popular format for representing and working with graph/network data. `vcount(g)` shows that the number of vertices (i.e. terms) is 3976. `ecount(g)` shows that the number of edges is 201792.

Naturally, this would not be an easy network to interpret. Therefore, we first filter on the most important vertices and edges. There are several methods to do so (see e.g., [Leydesdorff & Welbers, 2011]{http://arxiv.org/abs/1011.5209}). Here we use backbone extraction, which is a relatively new method (see [Kim & Kim, 2015]{http://jcom.sissa.it/archive/14/01/JCOM_1401_2015_A01}. Essentially, this method filters out edges that are not significant based on an alpha value, which can be interpreted similar to a p-value. To filter out vertices, we lower the alpha to a point where only the specified number of vertices remains.

```
g_backbone = getBackboneNetwork(g, alpha=0.0001, max.vertices=100)
```

```
## Used cutoff alpha 3.98523848383456e-05 to keep number of vertices under 100
```

```
## (For the edges the threshold assigned in the alpha parameter is still used)
```

```
vcount(g_backbone)
```

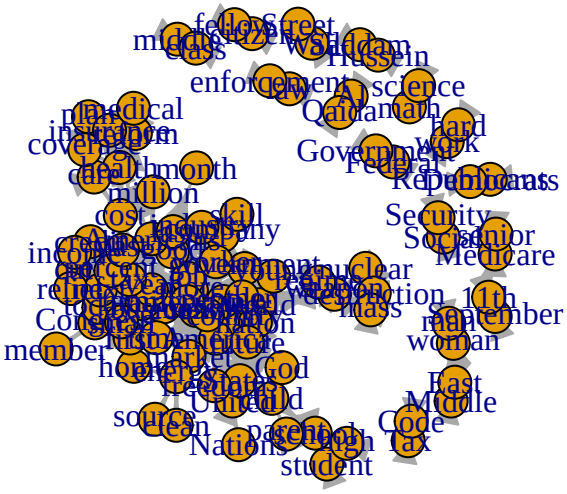
```
## [1] 100
```

```
ecount(g_backbone)
```

```
## [1] 255
```

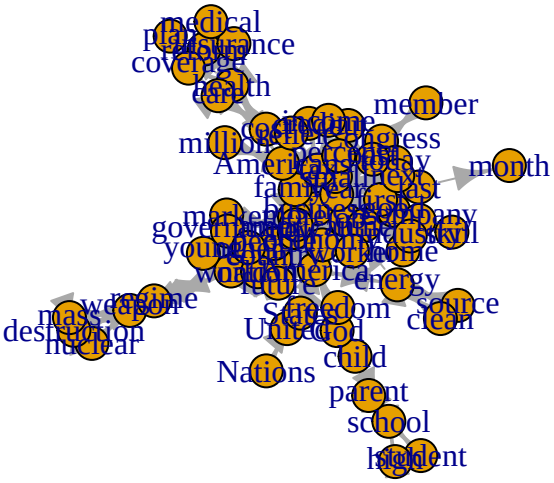
Now there are only 100 vertices and 255 edge left. This is a network we can interpret. Let's plot!

```
plot(g_backbone)
```



Nice, but still a bit messy. We can take some additional steps to focus the analysis and add additional information. First, we can look only at the largest connected component, thus ignoring small islands of terms such as `math` and `science`.

```
# select only largest connected component
g_backbone = decompose.graph(g_backbone, max.comps=1)[[1]]
plot(g_backbone)
```



Next, it would be interesting to take into account how often each term occured. This can be visualized by using the frequency of terms to set the sizes of the vertices. Also, we can use colors to indicate different clusters.

The output of the (windowed)coOccurenceNetwork function by default contains the vertex attribute `freq`, which can be used to set the vertex sizes. To find clusters, several community detection algorithms are available. To use this information for visualization some basic understanding of plotting igraph objects is required, which is out of the scope of this tutorial. We do provide a function named `setNetworkAttributes` which deals with these and some other visualization attributes.

```
V(g_backbone)$cluster = edge.betweenness.community(g_backbone)$membership

g_backbone = setNetworkAttributes(g_backbone, size_attribute=V(g_backbone)$freq,
  cluster_attribute=V(g_backbone)$cluster)
```

```
plot(g_backbone)
```

Now we have a more focused and informational visualization. We can for instance see several clusters that represent important talking points, such as the health care debate and the issue of nuclear weapons. Also, we see that America is at the center of discussions, in particular in context of economy and the job market.

Extracting Quantitative Information from networks

Computing network metrics

The igraph package contains numerous functions for computing network statistics. For example, this code computes the degree centrality (links per node) for the original semantic network:

```
c = centralization.degree(g)
degree = data.frame(node=V(g)$name, degree=c$res)
head(arrange(degree, -degree))
```

node	degree
America	2598
year	2510
people	2358
new	2180
country	2050
more	1998

Or you can get the most central nodes in the backgone network (using betweenness centrality)

```
c = centralization.betweenness(g_backbone)
centrality = data.frame(node=V(g_backbone)$name, centrality=c$res)
head(arrange(centrality, -centrality))
```

node	centrality
America	1588.5747
year	1433.3884
job	781.8806
tax	618.6967
world	592.8101
Americans	574.0363

Extracting relations

If we want to do more quantitative anlaysis of the network it can be useful to extract all relations as a data frame:

```
edges = as_data_frame(g, what="edges")
head(edges)
```

from	to	weight
task	unfinished	3
basic	unfinished	1
bargain	unfinished	1
country	unfinished	2
idea	unfinished	1
people	unfinished	1

The most frequent edges are also good candidates for collocation:

```
edges = arrange(edges, -weight)
head(edges)
```

from	to	weight
health	care	82
year	last	81
care	health	80
United	States	77
States	United	76
american	people	72

Extraecting co-occurrences per document

If you want to see in which documents an edge is contained, for example to add a time dimension to the network or to compare sources, you can set `output.per.context=T` in the original call. Since this will take a lot longer to run and generate a lot of output, we limit here to a sample of 10 paragraphs from Obama's speeches:

```
smp = sample(sotu.meta$id[sotu.meta$headline == "Barack Obama"], 10)
edges = with(sotu.tokens[sotu.tokens$aid %in% smp, ],
  windowedCoOccurenceNetwork(location=id, term=lemma, context=aid,
```

```
                      window.size=20, output.per.context = T))
head(edges)
```

x	y	context	weight
rubble	crisis	111551833	1
rubble	confidence	111551833	1
rubble	state	111551833	1
rubble	Union	111551833	1
rubble	stronger	111551833	1
crisis	rubble	111551833	1

Semnet for sentiment anlaysis

We can also use the word-windo approach for sentiment analysis. Suppose we would want to get the sentiment in phrases around 'Iraq' and 'terror'. First, we need to load a sentiment dictionary and for this exercise we will use a list to store the dictionary:

```
lexicon = readRDS("data/lexicon.rds")
dictionary = list(
  pos = lexicon$word1[lexicon$priorpolarity == "positive"],
  neg = lexicon$word1[lexicon$priorpolarity == "negative"],
  iraq = c("Iraq", "Iraqi" ),
  terror = c("terror", "terrorism", "terrorist"))
```

Now, we can use this to make a new 'concept' column that contains that concept and the sentiment values positive or negative

```
data(sotu)
sotu.tokens$concept = NA
for (concept in names(dictionary)) {
  sotu.tokens$concept[sotu.tokens$lemma %in% dictionary[[concept]]] = concept
}
table(sotu.tokens$concept)
```

```
##
##   iraq    neg    pos terror
##    109   3526   6972    182
```

We can use now get all windowed co-occurrences for these concepts using:

```
hits = windowedCoOccurenceNetwork(location=sotu.tokens$id, term=sotu.tokens$concept, context=sotu.tokens
                      window.size=20, output.per.context = T)
head(hits)
```

x	y	context	weight
neg	pos	111541965	1
neg	pos	111541965	1

x	y	context	weight
neg	pos	111541995	1
neg	pos	111541995	1
neg	pos	111541995	1
neg	pos	111541995	1

Now, we can compute a sentiment score and get the mean sentiment per context, excluding pos and neg itself:

```
hits$sentiment[hits$y == "pos"] = 1
hits$sentiment[hits$y == "neg"] = -1
hits = hits[!(hits$x %in% c("pos", "neg")), ]
library(reshape2)
sent = dcast(hits, context ~ x, value.var="sentiment", fun.aggregate = mean)
head(sent)
```

context	iraq	terror
111542025	NaN	1
111542114	NaN	0
111542119	NaN	0
111542189	0	NaN
111542284	1	NaN
111542285	NaN	0

Finally, let's merge that back with the metadata to get sentiment about Iraq and teror per president:

```
sent = merge(sotu.meta, sent, by.x="id", by.y="context")
aggregate(sent[c("iraq", "terror")], sent["headline"], mean, na.rm=T)
```

```
##         headline       iraq      terror
## 1   Barack Obama 0.05333333  0.13333333
## 2 George W. Bush 0.13530778 -0.06615646
```

So, both presidents are positive about (their policy in) Iraq, but while Bush is negative about terror, Obama is actually positive (presumably mostly talking about his efforts to contain it).