Query tutorial

Kasper Welbers

March 11, 2016

Searching and coding tokens

This document explains how semnet can be used to search and code tokens in a tokenlist. Before we start, we first need to create or load a tokenlist. If this does not ring a bell, please consult the create_tokenlist.Rmd manual first for detailed instructions.

```
library(semnet)
text = c('Renewable fuel is better than fossil fuels!',
           'A fueled debate about fuel',
           'Mark_Rutte is simply Rutte')
tokens = asTokenlist(text)
tokens
```

doc_id	position	word
1	1	renewable
1	2	fuel
1	3	is
1	4	better
1	5	than
1	6	fossil
1	7	fuels
1	8	!
2	1	a
2	2	fueled
2	3	debate
2	4	about
2	5	fuel
3	1	mark_rutte
3	2	is
3	3	simply
3	4	rutte

Above we see a simple tokenlist. For various purposes, it can be usefull to search tokens or code tokens (i.e. rows) into categories. For instance, if we are interested in political parties, we might want to find all politicians within that party and add them to a single category. This also enables us to analyze semantic networks at a higher aggregated level, using concepts or entities instead of individual words that refer to these concepts.

To do so, we can use queries consisting of multiple words and regular expressions. This is often referred to as a dictionary (see, for instance, quanteda::applyDictionary()). However, a limitation of this dictionary approach is that it is not possible to specify conditions based on surrounding words. For instance, if we are interested in the concept "Renewable fuel", we want to code the word "fuel" only if the surrounding words indicate that it refers to renewable fuel, and now fossil fuels.

The semnet package therefore offers an extension of the dictionary approach, where dictionary entries can

also contain Boolean like conditions (OR, AND and NOT, and the use of parentheses), that can also take word distance into account. Since this makes it more complex than common dictionaries, we instead refer to multiple queries as a "codebook".

A semnet query consists of two elements: an indicator and a condition. This tutorial first discusses both elements in detail, and introduces the searchQuery function to use individual queries. It concludes with a summary. For the use of multiple queries, please consult codebook_tutorial.Rmd.

Indicators

The indicator is the actual word, or set of words, that has to match a word in the tokenlist. For example, the word "fuel"

```
hits = searchQuery(tokens, indicator = 'fuel')
hits
```

	i	doc_id	position	code	word
2	2	1	2		fuel
13	13	2	5		fuel

The output of searchQuery shows: *Which rows in tokens are found (i)* The document id (doc_id) and word position (position) *The code assigned to the query (code). This is particularly usefull if multiple queries are used with searchQueries().* The exact word matched by the query

The reportSummary() function can be used to evaluate hits. This shows the number of hits, the frequencies of all words that match the query, and 10 random sentences in which the word occurs.

```
reportSummary(tokens, hits, kwic_nwords = 100)
```

```
## $hitcount
## hits docs
##    2    2
##
## $tokenfreq
##   code word freq doc_freq doc_pct
## 1      fuel    2        2     100
##
## $keywordIC
##    code doc_id position
## 13           2        5
## 2            1        2
##                                                 kwic
## 13                  ...a fueled debate about [fuel]...
## 2  ...renewable [fuel] is better than fossil fuels!...
```

OR statements

We can also give multiple indicators by using OR statements. For convenience, an empty space between words is also interpreted as an OR statement. Thus, "fuel OR fuels" is the same as "fuel fuels".

```
searchQuery(tokens, indicator = 'fuel fuels')
```

	i	doc_id	position	code	word
2	2	1	2		fuel
7	7	1	7		fuels
13	13	2	5		fuel

wildcards

Often it is usefull to use wildcards in words, for instance to make one term to find "fuel" and "fuels". If the ? symbol is used in a word, any single character can be used at this place (thus also at least one character). If the * symbol is used, any number of characters can be used at this place (including no characters at all)

```
searchQuery(tokens, indicator = 'fuel*')
```

	i	doc_id	position	code	word
2	2	1	2		fuel
7	7	1	7		fuels
10	10	2	2		fueled
13	13	2	5		fuel

Conditions

Sometimes, we only want to find a word if certain conditions are satisfied. For instance, if we want to look specifically where renewable fuels are discussed, we do not want to find fossil fuels. Therefore, a condition can be given for a query.

A condition should be interpreted as an AND statement. Thus, the indicator is only accepted if the condition is TRUE. In the following example, we set the condition that "renewable" OR "green" OR "clean" also need to occur within the same document as the indicator.

```
hits = searchQuery(tokens, indicator = 'fuel*', condition = 'renewable green clean')
reportSummary(tokens, hits)$keywordIC
```

	code	doc_id	position	kwic
2		1	2	... renewable [fuel] is better than fossil fuels!...
7		1	7	... renewable fuel is better than fossil [fuels]!...

Even more specifically, it can also be set that the condition has to occur within a given word distance of the indicator term (which we refer to as a window). This word window can be set in two ways. One is to use the default.window parameter.

```
hits = searchQuery(tokens, indicator = 'fuel*', condition = 'renewable green clean',
                   default.window = 2)
reportSummary(tokens, hits)$keywordIC
```

	code	doc_id	position	kwic
2		1	2	... renewable [fuel] is better than fossil fuels!...

The other way is to set the word window for specific words, by using the ~ symbol. For example, "renewable~3 green~5" would mean that either "renewable" has to occur within 3 words of the indicator, OR "green" has to occur within 5 words.

Finally, if a default.window is used, then the ~ symbol can also be used to ignore this window for specific words, by using word~d (where 'd' stands for document).

Complex conditions

Conditions can also contain more complex boolean operators. They can contain AND statements, and parentheses can be used. For example, the condition "(united AND states) OR us" would be true if either "us" occurs, OR if "united" AND "states" occur (within the given word distance).

Conditions can contain NOT statements. For example, "fuel* NOT fuell" matches al words starting with fuel, except fuell (a baseball player, apparently). Can also be used in combination with a single * to refer to anything, as in: "* NOT fuell".

Conditions can also use the ? and * wildcards, as described above

In some cases, we only want the condition to be satisfied at least once in an article. For instance, consider the second article in the example tokens: "Mark_Rutte is simply Rutte". Here, we can assume that the second Rutte is Mark Rutte. It is common that a person is first mentioned by full name, and then only by last name. To take this into account, we can use the "condition_once" parameter, which means that if the condition is satisfied once in an article, all occurences of the indicator are counted.

```
searchQuery(tokens, indicator = 'rutte', condition = 'mark~2')
```

	i	doc_id	position	code	word
14	14	3	1		mark rutte

```
searchQuery(tokens, indicator = 'rutte', condition = 'mark~2', condition_once = T)
```

	i	doc_id	position	code	word
14	14	3	1		mark rutte
17	17	3	4		rutte

Summary

A query consists of an indicator, and optionally a condition.

- *The indicator*
  - is the actual text that has to be found in the token
  - can contain multiple words with OR statement (and empty spaces are also considered OR statements)

  - CANNOT contain AND or NOT statements (this is what the condition is for)
  - accepts the ? wildcard, which means that any single character can be used in this place
  - accepts the * wildcard, which means that any number of characters can be used in this place

- *The condition*
  - has to be TRUE for the indicator to be accepted. Thus, if a condition is given, the query can be interpreted as: indicator AND condition
  - can contain complex boolean statements, using AND, OR and NOT statements, and using parentheses
  - accepts the ? and * wildcards
  - can be specified for a maximum word distance of the indicator. The terms in the condition are looked up within this word distance. The default word distance can be given with the default.window parameter. More specifically, individual terms can be given a custom word distance using the ~ symbol, where "word~50" means that "word" is looked up within 50 words of the indicator. If a default.window is used, it is also possible to ignore the word distance for specific terms by using word~d (where d stands for document).

- *Parameters*:
  - default.window -> determines the default word distance of the condition terms to the indicator (thus, if no specific word distance is set with the ~ symbol)
  - condition_once -> if TRUE, then if the condition is satisfied at least once in an article, all occurences of the indicator are accepted.