Visualizing tokens using the reader template

Kasper Welbers

April 25, 2017

This quick and dirty howto shows how to visualizate tokens as full texts using the reader template. Given tokens in a data.frame format, the create_reader() function creates an html file in which the tokens are pasted back to texts. Furthermore, it supports ways to highlight or color scale tokens.

To start, we open tokenvis and load the SOTU (state of the union) data.

```
library(tokenvis)
data(sotu)
```

The SOTU data consists of two data.frames: the tokens and the document meta data. Both data.frames have a "doc_id"" column, to match the tokens to the document meta. The only required columns are the doc_id and token column in the tokens data.frame. The meta data can optionally be used as additional information in the html reader.

```
head(tokens)
```

```
##      doc_id token_index      token
## 1 111541965           1         It
## 2 111541965           2         is
## 3 111541965           3        our
## 4 111541965           4 unfinished
## 5 111541965           5       task
## 6 111541965           6         to
```

```
head(meta)
```

```
##      doc_id       date     party    president
## 1 111541965 2013-02-12 Democrats Barack Obama
## 2 111541995 2013-02-12 Democrats Barack Obama
## 3 111542001 2013-02-12 Democrats Barack Obama
## 4 111542006 2013-02-12 Democrats Barack Obama
## 5 111542013 2013-02-12 Democrats Barack Obama
## 6 111542018 2013-02-12 Democrats Barack Obama
```

First, we create a simple reader using the create_reader function. By default, the function stores the html as a temporary file. The function returns the url to this location, which can then conveniently be used to open this url in the browser using the browseURL function.

```
url = create_reader(tokens, meta)
```

```
## Writing html to /tmp/RtmpPajBce/tokenvis_839eb5afe1.html
```

```
browseURL(url)
```

To color or otherwise annotate the words in the reader, we can first add html tags to the tokens. This can be done manually using the tag_tokens() function and its support functions (these will be explained in another manual). For the sake of convenience, there are several standard wrappers for coloring words. Here we demonstrate the highlighted_reader and colorscaled_reader functions.

The highlighted_reader function simply takes an additional argument, value. If value is a logical vector, it specifies which tokens to highlight. Alternatively, if it is a numeric vector with values between 0 and 1, it specifies how strongly words are highlighted (tokens with a value of 0 or NA will not be tagged). For this demo we highlight words based on the number of characters.

```r
highlight = nchar(as.character(tokens$token))
highlight = highlight / max(highlight)
url = highlighted_reader(tokens, value = highlight, meta)
```

```
## Writing html to /tmp/RtmpPajBce/tokenvis_839984ca3c.html
```

```r
browseURL(url)
```

Next, the colorscaled_reader function can similarly color words, but instead of highlighting it uses a scale ranging from -1 to 1. This is for instance usefull to visualize sentiment words or the results of a wordscaling analysis. For now, we simply use the number of characters again.

```r
scale = highlight*2 - 1
scale[abs(scale) < 0.4] = NA  ## add a threshold, for illustration
url = colorscaled_reader(tokens, value = scale, meta=meta)
```

```
## Writing html to /tmp/RtmpPajBce/tokenvis_8394b2356c9.html
```

```r
browseURL(url)
```