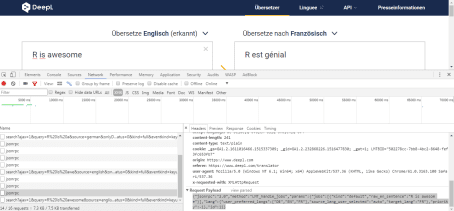
One of the great features of R is the possibility to quickly access web-services. While some companies have the habit and policy to document their APIs, there is still a large chunk of undocumented but great web-services that help the regular data scientist.

In the following short post, I will show how we can turn a simple web-serivce in a nice R-function.  
The example I am going to use is the linguee translation service: [DeepL](http://flovv.github.io/Accessing_a_web_api/www.deepl.com).  
Just as google translate, Deepl features a simple text field. When a user types in text, the translation appears in a second textbox. Users can choose between the languages.

In order to see how the service works in the backend, let’s have a quick look at the network traffic.  
For that we open the browser’s developer tools and jump to the network tab. Next, we type in a sentence and see which requests (XHR) are made. The interface repeatedly sends JSON requests to the following endpoint: “https://www.deepl.com/jsonrpc”.



Looking at a single request we can quickly identify the parameters that we typed in (grey area, in the lower right corner). We copy these in r and assign them to a variable.

str <- '{"jsonrpc":"2.0","method":"LMT\_handle\_jobs","params":{"jobs":[{"kind":"default","raw\_en\_sentence":"R is awesome"}],"lang":{"user\_preferred\_langs":["DE","EN","FR"],"source\_lang\_user\_selected":"auto","target\_lang":"FR"},"priority":-1},"id":11}'

Using a service to format the json (e.g. https://jsonformatter.curiousconcept.com/) we can turn the blob in a well readable json file. Next, we convert the JSON string in a R object (a nested list) by using a simple JSON to R language translation:

require(stringr)  
str1 <- str\_replace\_all(str, ":", "=")  
str2 <- str\_replace\_all(str1, "\\[", "list(")  
str3 <- str\_replace\_all(str2, "\\]", ")" )  
str4 <- str\_replace\_all(str3, "\\{", "list(" )  
str5 <- str\_replace\_all(str4, "\\}", ")" )  
   
eval(parse(text=str5))

Finally, we evaluate the string as R-code, this gives us the DeepL web-services’ parameters as an R nested list.  
All we have to do now is wrap the parameters in a R function and use variables to change the important ones:

require(rjson)  
require(httr)  
deepLTranslate <- function(text="R is awesome", from\_lang="EN", to\_lang="DE"){  
   
 BASE\_URL = 'https://www.deepl.com/jsonrpc'  
 JSONRPC\_VERSION = '2.0'  
 DEEPL\_METHOD = 'LMT\_handle\_jobs'  
   
 params =list('jsonrpc'= JSONRPC\_VERSION,   
 'method'= DEEPL\_METHOD,   
 params= list(  
 'jobs'=list(list('kind'= "defaut", 'raw\_en\_sentence'= text)),  
 'lang'=list(  
 'user\_preferred\_langs'=list(from\_lang,to\_lang),  
 'source\_lang\_user\_selected'= from\_lang, 'target\_lang'=to\_lang)  
 )   
 )  
   
 res <- POST(BASE\_URL,body = toJSON(params))  
   
 co <- content(res, "text")  
   
 if(res$status\_code ==200){  
 return(fromJSON(co))  
 }  
 else{  
 return(co)  
 }  
   
}  
   
#### excute the function with defaults ...  
deepLTranslate()

## $id  
## [1] 0  
##   
## $jsonrpc  
## [1] "2.0"  
##   
## $result  
## $result$source\_lang  
## [1] "EN"  
##   
## $result$source\_lang\_is\_confident  
## [1] 0  
##   
## $result$target\_lang  
## [1] "DE"  
##   
## $result$translations  
## $result$translations[[1]]  
## $result$translations[[1]]$beams  
## $result$translations[[1]]$beams[[1]]  
## $result$translations[[1]]$beams[[1]]$num\_symbols  
## [1] 5  
##   
## $result$translations[[1]]$beams[[1]]$postprocessed\_sentence  
## [1] "R ist fantastisch"  
##   
## $result$translations[[1]]$beams[[1]]$score  
## [1] -5000.6  
##   
## $result$translations[[1]]$beams[[1]]$totalLogProb  
## [1] -4.37026  
##   
##   
## $result$translations[[1]]$beams[[2]]  
## $result$translations[[1]]$beams[[2]]$num\_symbols  
## [1] 5  
##   
## $result$translations[[1]]$beams[[2]]$postprocessed\_sentence  
## [1] "R ist großartig"  
##   
## $result$translations[[1]]$beams[[2]]$score  
## [1] -5000.64  
##   
## $result$translations[[1]]$beams[[2]]$totalLogProb  
## [1] -4.6875  
##   
##   
## $result$translations[[1]]$beams[[3]]  
## $result$translations[[1]]$beams[[3]]$num\_symbols  
## [1] 6  
##   
## $result$translations[[1]]$beams[[3]]$postprocessed\_sentence  
## [1] "R ist fantastisch."  
##   
## $result$translations[[1]]$beams[[3]]$score  
## [1] -5000.67  
##   
## $result$translations[[1]]$beams[[3]]$totalLogProb  
## [1] -5.57148  
##   
##   
## $result$translations[[1]]$beams[[4]]  
## $result$translations[[1]]$beams[[4]]$num\_symbols  
## [1] 6  
##   
## $result$translations[[1]]$beams[[4]]$postprocessed\_sentence  
## [1] "R ist großartig."  
##   
## $result$translations[[1]]$beams[[4]]$score  
## [1] -5000.72  
##   
## $result$translations[[1]]$beams[[4]]$totalLogProb  
## [1] -6.03852  
##   
##   
##   
## $result$translations[[1]]$timeAfterPreprocessing  
## [1] 0  
##   
## $result$translations[[1]]$timeReceivedFromEndpoint  
## [1] 311  
##   
## $result$translations[[1]]$timeSentToEndpoint  
## [1] 0  
##   
## $result$translations[[1]]$total\_time\_endpoint  
## [1] 1

If you are looking for other translation services have a look at the translate or translateR packages.