(Syntactic) Parsing in R

Martin Schweinberger

December 28, 2017

This post will exemplify how to syntactically parse a corpus with R (the function is available via
`http://martinschweinberger.de/docs/scripts/paRsing.r`).
Syntactic Parsing is a form of annotating text in which POS tags are assigned to lexical items and then lexical items are grouped together in phrasal constituents. Syntactic parsing is thus an extension of POS tagging as syntactic parsing requires POS tagging. This post will not go into the theoretical background and various approaches to syntactic parsing – syntactic parsing is quite complex both in terms of theory and practical implementation - but it will simply show how you can use R to parse some text based on the Apache OpenNLP Maxent Parser.

In R we can syntactically parse large amounts of text using the openNLP package, which also requires the NLP package and installing the models on which the openNLP package works – you can find more information on the openNLP package and how it works here:
`http://cran.r-project.org/web/packages/openNLP/openNLP.pdf`.
The openNLP package uses the Apache OpenNLP Maxent Parser which is a trained parser, which works in two steps. In a first step, the included POS tagger assigns POS tags based on the probability of what the correct POS tag is – the POS tag with the highest probability is selected. In a next step, the lexical items are grouped together into phrasal and finally clausal constituents.

Unforunately, there is a real issue when R interfaces with Java (which is what we do when we use the openNLP package) – R will report an error:

```
java.lang.OutOfMemoryError:  Java heap space
```

This error indicates that the memory that is taken up by the task is exploding. There is a way around it which is however not very nice: you need to close R and then run your function again (the command `gc()` is also meant to prevent the memory from becoming too big but it does not seem to work properly). So, if this error occurs, close R, open it again, then call the function, and apply it to some text.

Below is an example of how you can implement syntactic parsing in R.

```
# write function
paRsing <- function(path){
  require("NLP")
  require("openNLP")
  require("openNLPmodels.en")
  require("stringr")
  corpus.files = list.files(path = path, pattern = NULL, all.
     files = T,
    full.names = T, recursive = T, ignore.case = T, include.
       dirs = T)
  corpus.tmp <- lapply(corpus.files, function(x) {
    scan(x, what = "char", sep = "\t", quiet = T) }  )
  corpus.tmp <- lapply(corpus.tmp, function(x){
    x <- paste(x, collapse = " ")  }  )
  corpus.tmp <- lapply(corpus.tmp, function(x) {
    x <- enc2utf8(x)  }  )
  corpus.tmp <- gsub(" {2,}", " ", corpus.tmp)
  corpus.tmp <- str_trim(corpus.tmp, side = "both")
  sent_token_annotator <- Maxent_Sent_Token_Annotator()
  word_token_annotator <- Maxent_Word_Token_Annotator()
  parse_annotator <- Parse_Annotator()
  Corpus <- lapply(corpus.tmp, function(x){
    x <- as.String(x)  }  )
  lapply(Corpus, function(x){
    annotated <- annotate(x, list(sent_token_annotator,
       word_token_annotator))
# Compute the parse annotations only.
    parsed <- parse_annotator(x, annotated)
# Extract the formatted parse trees.
    parsedtexts <- sapply(parsed$features, '[[', "parse")
# Read into NLP Tree objects.
    parsetrees <- lapply(parsedtexts, Tree_parse)
    gc()
        return(list(parsedtexts, parsetrees))
 }  )
  }


##########################################################
# test the function
```

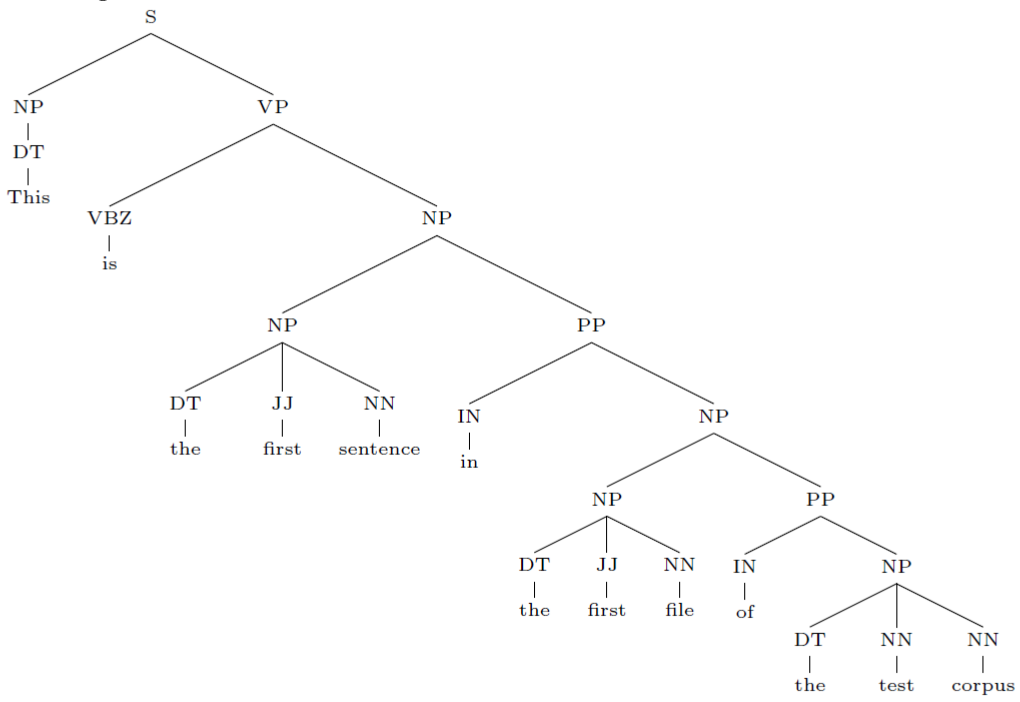
```
parsetest <- paRsing(path = "C:\\03-MyProjects\\PosTagging\\
    TestCorpus")
```

The parsed first sentence is shown on the R GUI as follows:

```
# inspect the first sentence
parsetest[[1]][[1]][[1]]

#[1] "(TOP (S (NP (DT This)) (VP (VBZ is) (NP (NP (DT the) (
    JJ first) (NN sentence)) (PP (IN in) (NP (NP (DT the) (JJ
    first) (NN file)) (PP (IN of) (NP (DT the) (NN test) (NN
    corpus)))))))(. .)))"
```

You can use the output to find e.g. all noun phrases in a text or to create fancy tree diagrams like the one here:



I hope this helps and I will also be posting some updates and show what parsing can be used for.