Part-Of-Speech Tagging with R

Martin Schweinberger

June 24, 2016

Introduction

This post[1] exemplifies how to add Part–of-Speech annotation (POS–tags) to corpus data with `R`. Part-of-Speech tagging, or POS–tagging, is a form of annotating text during which Part-of-Speech tags are assigned to character strings (these represent mostly words, of course, but also encompass punctuation marks and other elements). This means that POS–tagging is one specific type of annotation, i.e. adding information to data (either by directly adding information to the data itself or by storing information in e.g. a list which is linked to the data). It is important to note that annotation encompasses various types of information such as pauses, overlap, etc. POS–tagging is just one of these many ways in which corpus data can be "enriched".

Parts-of-speech, or word categories, refer to the grammatical nature or category of a lexical item, e.g. in the sentence "Jane likes the girl" each lexical item can be classified according to whether it belongs to the group of determiners, verbs, nouns, etc. When POS–tagged, the example sentence could look like the example below.

(1) Jane\NNP likes\VBZ the\DT girl\NN

In the example above, `NNP` stands for proper noun (singular), `VBZ` stands for 3rd person singular present tense verb, `DT` for determiner, and `NN` for noun (singular or mass). The POS tags used by the `openNLP` package are the *Penn English Treebank* POS tags – here is a list of these tags and what they stand for:

[1] Update: Joseph Flanagan has found a solution to the memory overkill in the code. The post was updated including his suggestion to place the annotators outside the loop on 2015-06-08.

Part–of–Speech Tag	Part–of–Speech category
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non–3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh–determiner
WP	Wh–pronoun
WP$	Possessive wh–pronoun
WRB	Wh–adverb

Assigning these POS tags to words appears to be rather straight forward. However, POS tagging is quite complex and there are various ways by which a computer can be trained to assign POS tags. For example, one could use orthographic or morphological information to devise rules such as...

(2) If a word end with {ment} assign the POS tag NN (for common noun)

(3) If a word does not occur at the beginning of a sentence but is capitalized, assign the POS tag NNP (for proper noun)

Using such rules has the disadvantage that POS tags can only be assigned to a relatively small number of words as most words will be ambiguous – think of the similarity of the English plural and the English past tense morpheme, for instance, which are orthographically identical.

Another option would be to use a dictionary in which each word is assigned a certain POS tag and a program could assign the POS tag if the word occurs in a given text. This procedure has the disadvantage that most words belong to more than one word class and POS tagging would thus have to rely on additional information.

The problem of words that belong to more than one word class can partly be remedied by including contextual information such as...

(4) If the previous word is a determiner and the subsequent word is a common noun, assign the POS tag `JJ` (for a common adjective)

This procedure works quite well but there are still better options.

The best way to POS tag a text is to create a manually annotated training set which resembles the language variety at hand. Based on the frequency of the association between a given word and the POS tags it is assigned in the training data, it is possible to tag a word with the POS tag that is most often assigned to the given word in the training data.

All of the above methods can and should be optimized by combining them and additionally including POS–n–grams, i.e. determining a POS tag of an unknown word based on which sequence of POS tags is most similar to the sequence at hand and also most common in the training data.

This introduction is extremely superficial and only intends to scratch some of the basic procedures that POS tagging relies on. The interested reader is referred to introductions on machine learning and POS tagging such as e.g. `https://class.coursera.org/nlp/lecture/149`.

Part-Of-Speech Tagging with R using the openNLP package

In `R` we can POS–tag large amounts of text by various means. This section explores POS tagging using the openNLP package. Using the openNLP library for POS tagging works particularly well when the aim is to POS tag newspaper texts as the openNLP library implements the Apache OpenNLP Maxent Part of Speech tagger and it comes with pre-trained models. Ideally, POS taggers should be trained on data resembling the data to be POS tagged. However, I do not know how to traind the Apache openNLP POS tagger via `R` and it would be great if someone would provide a tutorial on how to do that.

Using pre-trained models has the advantage that we do not need to train the POS tagger ourselves. However, it also means that one has to rely on models trained on data that may not really resemble the data a at hand. This implies that using it for texts that differ from newspaper texts, i.e. the language the models have been trained on, does not work as well, as the model applies the probabilities of newspaper language to the language variety at hand.

POS tagging with the openNLP 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 at this site:

http://cran.r-project.org/web/packages/openNLP/openNLP.pdf. The openNLP package uses the Apache OpenNLP Maxent Part of Speech tagger which is a trained POS tagger, that assigns POS tags based on the probability of what the correct POS tag is – the POS tag with the highest probability is selected.

Below is an example of how you can implement POS tagging in R. In a first step, we start our script by providing a short introduction with title date and short description and continue by removing objects from the existing work space.

```
##########################################################
### --- Part-of-Speech tagging and syntactic parsing with R
### --- Title: Part-of-Speech tagging with R
### --- Author: Martin Schweinberger
### --- This script aims at an automated approach
### --- to POS tagging a sample corpus.
##########################################################
# Remove all lists from the current workspace
rm(list=ls(all=T))
```

After setting up our script, we install all libraries that are either required or useful for POS tagging corpus data. In case you have not already installed these libraries, install them using the install.packages function and remove the # to activate the commands and install the packages. It is crucial to install the openNLP models if you want to use this library for POS tagging. The models are pre-trained POS taggers available for various languages – we are only using a model for English in this example, though.

```
# Install packages we need or which may be useful
# (to activate just delete the #)
#install.packages("openNLPmodels.en", repos = "http://datacube.wu.ac.at/", type = "source")
#install.packages("openNLP")
#install.packages("NLP")
### additional packages
#install.packages("tm")
#install.packages("stringr")
#install.packages("gsubfn")
#install.packages("plyr")
# to install openNLPmodels, please download an install
# the packages/models directly from
# http://datacube.wu.ac.at/.
# To install these packages/models, simply enter
#install.packages("foo", repos = "http://datacube.wu.ac.at/", type = "source")
# into your R console. E.g. enter:
```

```r
#install.packages("openNLPmodels.en", repos = "http://datacube.wu.ac.at/", type = "source")
# to install the file "openNLPmodels.en_1.5-1.tar.gz"
```

If the libraries are already installed, they need to be activated using the `library` function.

```r
# activate packages
library(NLP)
library(openNLP)
library(openNLPmodels.en)
library(tm)
library(stringr)
library(gsubfn)
library(plyr)
```

After installing and activating the libraries and models, we set the paths to the data and prepare the data for POS tagging.

```r
# specify path of corpus
pathname <- "C:\\03-MyProjects\\PosTagging\\TestCorpus"
# choose files
corpus.files = list.files(path = pathname, pattern = NULL,
 all.files = T, full.names = T, recursive = T,
 ignore.case = T, include.dirs = T)
# load and unlist corpus
corpus.tmp <- lapply(corpus.files, function(x) {
 scan(x, what = "char", sep = "\t", quiet = T) } )
# Paste all elements of the corpus together
corpus.tmp <- lapply(corpus.tmp, function(x){
 x <- paste(x, collapse = " ") } )
# Clean corpus
corpus.tmp <- lapply(corpus.tmp, function(x) {
 x <- enc2utf8(x) } )
corpus.tmp <- gsub(" {2,}", " ", corpus.tmp)
# remove spaces at beginning and end of strings
corpus.tmp <- str_trim(corpus.tmp, side = "both")
# convert corpus files into strings
Corpus <- lapply(corpus.tmp, function(x){
 x <- as.String(x) } )
```

Once the data is cleaned, we may start tagging by applying the POS tagger to the data.

```r
# apply annotators to Corpus
```

```
Corpus.tagged <- lapply(Corpus, function(x){
 sent_token_annotator <- Maxent_Sent_Token_Annotator()
 word_token_annotator <- Maxent_Word_Token_Annotator()
 pos_tag_annotator <- Maxent_POS_Tag_Annotator()
 y1 <- annotate(x, list(sent_token_annotator, word_token_annotator))
 y2 <- annotate(x, pos_tag_annotator, y1)
# y3 <- annotate(x, Maxent_POS_Tag_Annotator(probs = TRUE), y1)
 y2w <- subset(y2, type == "word")
 tags <- sapply(y2w$features, '[[', "POS")
 r1 <- sprintf("%s/%s", x[y2w], tags)
 r2 <- paste(r1, collapse = " ")
 return(r2) } )
```

It is now possible to inspect the results by entering the name of the POS tagged object.

```
# inspect results
Corpus.tagged
```

The output produced by `R` is displayed below.

```
>[[1]]
>[1]    ”This/DT is/VBZ the/DT first/JJ sentence/NN in/IN the/DT first/JJ file/NN of/IN the/DT test/NN corpus/NN ./. This/DT is/VBZ a/DT second/JJ sentence/NN in/IN the/DT test/NN corpus/NN but/CC I/PRP am/VBP too/RB lazy/JJ to/TO write/VB much/RB more/RBR so/RB this/DT has/VBZ to/TO suffice/VB ./. well/RB ,/, one/CD more/JJR sentence/NN should/MD do/VB ./.”

>[[2]]
>[1]    ”This/DT is/VBZ a/DT second/JJ file/NN with/IN some/DT sample/NN content/NN ./. It/PRP will/MD be/VB used/VBN to/TO test/VB a/DT part-of-speech/NN tagger/NN in/IN R./NNP I/PRP dont/VBP really/RB know/VB if/IN it/PRP works/VBZ but/CC I/PRP definitely/RB hope/VBP so/RB ./.”

>[[3]]
>[1]    ”Finally/RB ,/, this/DT is/VBZ the/DT last/JJ file/NN of/IN the/DT test/NN corpus/NN and/CC I/PRP really/RB dont/VBP want/VB to/TO write/VB a/DT lot/NN more/RBR ./. Since/IN I/PRP am/VBP quite/RB lazy/JJ ,/, this/DT is/VBZ the/DT last/JJ sentence/NN in/IN my/PRP$ tiny/JJ test/NN corpus/NN ./.”
```

It is preferable to write a function to perform the POS tagging automatically rather than running all the lines of code semi-manually. The code below represents just that: a little function which POS tags corpus data. The function takes the path to the directory in which the corpus is located as an argument.

```
POStag <- function(path = path){
 require("NLP")
 require("openNLP")
 require("openNLPmodels.en")
 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")
 Corpus <- lapply(corpus.tmp, function(x){
 x <- as.String(x) } )
 sent_token_annotator <- Maxent_Sent_Token_Annotator()
 word_token_annotator <- Maxent_Word_Token_Annotator()
 pos_tag_annotator <- Maxent_POS_Tag_Annotator()
 lapply(Corpus, function(x){
 y1 <- annotate(x, list(sent_token_annotator,
 word_token_annotator))
 y2<- annotate(x, pos_tag_annotator, y1)
# y3 <- annotate(x, Maxent_POS_Tag_Annotator(probs = TRUE),
   y1)
 y2w <- subset(y2, type == "word")
 tags <- sapply(y2w$features, ’[[’, "POS")
 r1 <- sprintf("%s/%s", x[y2w], tags)
 r2 <- paste(r1, collapse = " ")
 return(r2) } )
 }
```

The function will be tested by applying it to a small test corpus.

```
# test the function
POStag(path = "C:\\03-MyProjects\\PosTagging\\TestCorpus")
```

The output of the function is displayed below.

```
>[[1]]
>[1]      ”This/DT is/VBZ the/DT first/JJ sentence/NN in/IN the/DT first/JJ file/NN
          of/IN the/DT test/NN corpus/NN ./. This/DT is/VBZ a/DT second/JJ sen-
          tence/NN in/IN the/DT test/NN corpus/NN but/CC I/PRP am/VBP too/RB
          lazy/JJ to/TO write/VB much/RB more/RBR so/RB this/DT has/VBZ
          to/TO suffice/VB ./. well/RB ,/, one/CD more/JJR sentence/NN should/MD
          do/VB ./.”

>[[2]]
>[1]      ”This/DT is/VBZ a/DT second/JJ file/NN with/IN some/DT sample/NN
          content/NN ./. It/PRP will/MD be/VB used/VBN to/TO test/VB a/DT
          part-of-speech/NN tagger/NN in/IN R./NNP I/PRP dont/VBP really/RB
          know/VB if/IN it/PRP works/VBZ but/CC I/PRP definitely/RB hope/VBP
          so/RB ./.”

>[[3]]
>[1]      ”Finally/RB ,/, this/DT is/VBZ the/DT last/JJ file/NN of/IN the/DT
          test/NN corpus/NN and/CC I/PRP really/RB dont/VBP want/VB to/TO
          write/VB a/DT lot/NN more/RBR ./. Since/IN I/PRP am/VBP quite/RB
          lazy/JJ ,/, this/DT is/VBZ the/DT last/JJ sentence/NN in/IN my/PRP$
          tiny/JJ test/NN corpus/NN ./.”

Warnmeldungen:
1: In scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings, :
EOF in Zeichenkette
2: In scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,
EOF in Zeichenkette
```

The output shows that the function fulfills its purpose and automatically POS tags the corpus data in the specified directory. The warnings that are printed by R can be ignored because they merely inform that the last line is not empty but contains content. I hope this helps and I will also be posting some updates to include more useful examples.

Part-Of-Speech Tagging with R using the Tree Tagger

Another much handier way to add POS tags to texts is to use the koRpus library rather than the openNLP library. The koRpus library uses the TreeTagger (cf. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/) for POS tagging. In other words, the TreeTagger has to be installed prior to running the script below as it accesses the TreeTagger via R.

You can find the code for implementing the TreeTagger below. However, a word of warning is advisable: It can be quite tedious to implement the

TreeTagger in case you are running a Windows machine (as I do). Most of the issues were solved when I re-installed Java though. Last but not least, I simply implement the TreeTagger without training it! This is in fact not a good practice and should be avoided as I have no way of knowing how good the performance is or what I could do to improve its performance!

```
# POS tagging in R with koRpus
# activate library
library(koRpus)
# define pathname
pathname = "C:\\03-MyProjects\\PosTagging\\TestCorpus"
# perform POS tagging
text.tagged <- treetag("C:\\03-MyProjects\\PosTagging\\TestCorpus/text1.txt", treetagger="manual", lang="en",
 TT.options=list(path="C:\\TreeTagger", preset="en"))
```

The results can be inspected by calling the object in which the results are stored.

```
# inspect text.tagged
text.tagged@TT.res
```

```
Slot ”TT.res”:
     token      tag    lemma     lttr  wclass       desc                                                    stop  stem

1    This       DT     this      4     determiner   Determiner                                              NA    NA
2    is         VBZ    be        2     verb         Verb, 3rd person singular present of ”to be”            NA    NA
3    the        DT     the       3     determiner   Determiner                                              NA    NA
4    first      JJ     first     5     adjective    Adjective                                               NA    NA
5    sentence   NN     sentence  8     noun         Noun, singular or mass                                  NA    NA
6    in         IN     in        2     preposition  Preposition or subordinating conjunction                NA    NA
7    the        DT     the       3     determiner   Determiner                                              NA    NA
8    first      JJ     first     5     adjective    Adjective                                               NA    NA
9    file       NN     file      4     noun         Noun, singular or mass                                  NA    NA
10   of         IN     of        2     preposition  Preposition or subordinating conjunction                NA    NA
11   the        DT     the       3     determiner   Determiner                                              NA    NA
12   test       NN     test      4     noun         Noun, singular or mass                                  NA    NA
13   corpus     NN     corpus    6     noun         Noun, singular or mass                                  NA    NA
14   .          SENT   .         1     fullstop     Sentence ending punctuation                             NA    NA
15   This       DT     this      4     determiner   Determiner                                              NA    NA
16   is         VBZ    be        2     verb         Verb, 3rd person singular present of ”to be”            NA    NA
17   a          DT     a         1     determiner   Determiner                                              NA    NA
18   second     JJ     second    6     adjective    Adjective                                               NA    NA
19   sentence   NN     sentence  8     noun         Noun, singular or mass                                  NA    NA
20   in         IN     in        2     preposition  Preposition or subordinating conjunction                NA    NA
21   the        DT     the       3     determiner   Determiner                                              NA    NA
22   test       NN     test      4     noun         Noun, singular or mass                                  NA    NA
23   corpus     NN     corpus    6     noun         Noun, singular or mass                                  NA    NA
24   but        CC     but       3     conjunction  Coordinating conjunction                                NA    NA
25   I          PP     I         1     pronoun      Personal pronoun                                        NA    NA
26   am         VBP    be        2     verb         Verb, non-3rd person singular present of ”to be”        NA    NA
27   too        RB     too       3     adverb       Adverb                                                  NA    NA
28   lazy       JJ     lazy      4     adjective    Adjective                                               NA    NA
29   to         TO     to        2     to           to                                                      NA    NA
30   write      VV     write     5     verb         Verb, base form                                         NA    NA
31   much       RB     much      4     adverb       Adverb                                                  NA    NA
32   more       RBR    more      4     adverb       Adverb, comparative                                     NA    NA
33   so         RB     so        2     adverb       Adverb                                                  NA    NA
34   this       DT     this      4     determiner   Determiner                                              NA    NA
35   has        VHZ    have      3     verb         Verb, 3rd person singular present of ”to have”          NA    NA
36   to         TO     to        2     to           to                                                      NA    NA
37   suffice    VV     suffice   7     verb         Verb, base form                                         NA    NA
38   .          SENT   .         1     fullstop     Sentence ending punctuation                             NA    NA
39   well       RB     well      4     adverb       Adverb                                                  NA    NA
40   ,          ,      ,         1     comma        Comma                                                   NA    NA
41   one        CD     one       3     number       Cardinal number                                         NA    NA
42   more       JJR    more      4     adjective    Adjective, comparative                                  NA    NA
43   sentence   NN     sentence  8     noun         Noun, singular or mass                                  NA    NA
44   should     MD     should    6     modal        Modal                                                   NA    NA
45   do         VV     do        2     verb         Verb, base form                                         NA    NA
46   .          SENT   .         1     fullstop     Sentence ending punctuation                             NA    NA
```

I hoped this short tutorial might help you POS tag your own data with R.