I made a `parts-of-speech` directory to keep the code self-contained. In it are two files. The first is `partsofspeech.swift` (`swiftc` seems to dislike dashes in names of library code and I dislike underscores):

NOTE: I didn't change the @'s this time, so just ignore the incorrect Twitter links it created.

```
import NaturalLanguage
import CoreML

extension Array where Element == String {
  var SEXP: SEXP? {
    let charVec = Rf_protect(Rf_allocVector(SEXPTYPE(STRSXP), count))
    defer { Rf_unprotect(1) }
    for (idx, elem) in enumerated() { SET_STRING_ELT(charVec, idx,
Rf_mkChar(elem)) }
    return(charVec)
  }
}

@_cdecl("part_of_speech")
public func part_of_speech(_ x: SEXP) -> SEXP {

  let text = String(cString: R_CHAR(STRING_ELT(x, 0)))
  let tagger = NLTagger(tagSchemes: [.lexicalClass])

  tagger.string = text

  let options: NLTagger.Options = [.omitPunctuation, .omitWhitespace]

  var txts = [String]()
  var tags = [String]()

  tagger.enumerateTags(in: text.startIndex..
```

The other is bridge code that seems to be the same for every one of these (or could be) so I've just named it `swift-r-glue.h` (it's the same as the bridge code in the previous post):

```
#define USE_RINTERNALS

#include
#include

const char* R_CHAR(SEXP x);
```

Let's walk through the Swift code.

We need to two `import`s:

```
import NaturalLanguage
import CoreML
```

to make use of the NLP functionality provided by Apple.

The following extension to the String Array class:

```
extension Array where Element == String {
  var SEXP: SEXP? {
    let charVec = Rf_protect(Rf_allocVector(SEXPTYPE(STRSXP), count))
    defer { Rf_unprotect(1) }
    for (idx, elem) in enumerated() { SET_STRING_ELT(charVec, idx,
Rf_mkChar(elem)) }
    return(charVec)
  }
}
```

will reduce the amount of code we need to type later on to turn Swift String Arrays to R character vectors.

The start of the function:

```
@_cdecl
public func part_of_speech(_ x: SEXP) -> SEXP {
```

tells `swiftc` to make this a C-compatible call and notes that the function takes one parameter (in this case, it's expecting a length 1 character vector) and returns an R-compatible value (which will be a `list` that we'll turn into a `data.frame` in R just for brevity).

The following sets up our inputs and outputs:

```
  let text = String(cString: R_CHAR(STRING_ELT(x, 0)))
  let tagger = NLTagger(tagSchemes: [.lexicalClass])
```

```
  tagger.string = text

  let options: NLTagger.Options = [.omitPunctuation, .omitWhitespace]

  var txts = [String]()
  var tags = [String]()
```

We convert the passed-in parameter to a Swift String, initialize the NLP tagger, and setup two arrays to hold the results (sentence component in `txts` and the part of speech that component is in `tags`).

The following code is mostly straight from Apple and (inefficiently) populates the previous two arrays:

```
  tagger.enumerateTags(in: text.startIndex..
```

Finally, we use the Swift-R bridge to make a `list` much like one would in C:

```
  let out = Rf_protect(Rf_allocVector(SEXPTYPE(VECSXP), 2))
  SET_VECTOR_ELT(out, 0, txts.SEXP)
  SET_VECTOR_ELT(out, 1, tags.SEXP)
  Rf_unprotect(1)

  return(out!)
```

To get a shared library we can use from R, we just need to compile this like last time:

```
swiftc \
  -I /Library/Frameworks/R.framework/Headers \
  -F/Library/Frameworks \
  -framework R \
  -import-objc-header swift-r-glue.h \
  -emit-library \
  partsofspeech.swift
```

Let’s run that on some text! First, we’ll load the new shared library into R:

```
dyn.load("libpartsofspeech.dylib")
```

Next, we'll make a wrapper function to avoid messy `.Call(…)`s and to make a `data.frame`:

```
parts_of_speech <- function(x) {
  res <- .Call("part_of_speech", x)
  as.data.frame(stats::setNames(res, c("name", "tag")))
}
```

Finally, let's try this on some text!

```
tibble::as_tibble(
  parts_of_speech(paste0(c(
"The comm wasn't working. Feeling increasingly ridiculous, he pushed",
"the button for the 1MC channel several more times. Nothing. He
opened",
"his eyes and saw that all the lights on the panel were out. Then he",
"turned around and saw that the lights on the refrigerator and the",
"ovens were out. It wasn't just the coffeemaker; the entire galley
was",
"in open revolt. Holden looked at the ship name, Rocinante, newly",
"stenciled onto the galley wall, and said, Baby, why do you hurt me",
"when I love you so much?"
  ), collapse = " "))
)
## # A tibble: 92 x 2
##    name         tag
##
##  1 The          Determiner
##  2 comm         Noun
##  3 was          Verb
##  4 n't          Adverb
##  5 working      Verb
##  6 Feeling      Verb
##  7 increasingly Adverb
##  8 ridiculous   Adjective
##  9 he           Pronoun
## 10 pushed       Verb
## # … with 82 more rows
```

FIN

If you're playing along at home, try adding a function to this Swift file that uses Apple's entity tagger.