**Python & reticulate set-up**

The Python code below sets up a conda environment and installs relevant libraries, as well as the BERT transformer, *en\_core\_web\_trf*.

conda create -n poly1

source activate poly1

conda install -c conda-forge spacy

python -m spacy download en\_core\_web\_trf

conda install numpy scipy pandas

The R code below directs R to our Python environment and Python installation.

reticulate::use\_python("/usr/local/bin/python")

reticulate::use\_condaenv("poly1", "/home/jtimm/anaconda3/bin/conda")

**COCA**

The Corpus of Contemporary American English (COCA) is an absolutely lovely resource, and is one of [many corpora](https://www.english-corpora.org/) made available by the folks at BYU. Here, we utilize COCA to build a simple data set of further–farther example usages. I have copied/pasted from COCA’s online search interface – the data set includes ~500 contexts of usage per form.

library(tidyverse)

gw <- read.csv(paste0(ld, 'further-farther.csv'), sep = '\t')

gw$sent <- tolower(gsub("([[:punct:]])", " \\1 ", gw$text))

gw$sent <- gsub("^ \*|(?<= ) | \*$", "", gw$sent, perl = TRUE)

gw$count <- stringr::str\_count(gw$sent, 'further|farther')

gw0 <- subset(gw, count == 1)

The standard semantic distinction drawn between the two forms is physical versus metaphorical distance.

Some highlighting & sample data below.

fu <- '\\1 <span style="background-color:lightgreen">\\2</span> \\3'

fa <- '\\1 <span style="background-color:lightblue">\\2</span> \\3'

gw0$text <- gsub('(^.+)(further)(.+$)', fu, gw0$text, ignore.case = T)

gw0$text <- gsub('(^.+)(farther)(.+$)', fa, gw0$text, ignore.case = T)

gw0$text <- paste0('... ', gw0$text, ' ...')

set.seed(99)

gw0 %>% select(year, genre, text) %>% sample\_n(10) %>%

DT::datatable(rownames = F, escape = F,

options = list(dom = 't',

pageLength = 10,

scrollX = TRUE))

Lastly, we identify the location (ie, context position) of the target token within each context (as *token index*).

gw0$idx <- sapply(gsub(' (farther|further).\*$', '', gw0$sent, ignore.case = T),

function(x){

length(corpus::text\_tokens(x)[[1]]) })

**BERT & contextual embeddings**

Using BERT and spacy for computing contextual word embeddings is actually fairly straightforward.

Getting started, we pass our data set from R to Python via the r\_to\_py function.

df <- reticulate::r\_to\_py(gw0)

Then, from a Python console, we load the BERT transformer using spacy.

import spacy

nlp = spacy.load('en\_core\_web\_trf')

**The stretch of Python code below does all the work here**. The transformer computes a 768 dimension vector per token/sub-token comprising each context – then we extract the tensor for either further/farther using the token index. The resulting data structure is matrix-like, with each instantiation of further-farther represented in 768 dimensions.

def encode(sent, index):

doc = nlp(sent.lower())

tensor\_ix = doc.\_.trf\_data.align[index].data.flatten()

out\_dim = doc.\_.trf\_data.tensors[0].shape[-1]

tensor = doc.\_.trf\_data.tensors[0].reshape(-1, out\_dim)[tensor\_ix]

## tensor.\_\_len\_\_()

return tensor.mean(axis=0)

r.df["emb"] = r.df[["sent", "idx"]].apply(lambda x: encode(x[0], x[1]), axis = 1)

**tSNE**

To plot these contexts in two dimensions, we use tSNE to reduce the 768-dimension word embeddings to two. Via Python and numpy, we create a matrix-proper from the further-farther token embeddings extracted above.

import numpy as np

X, y = r.df["emb"].values, r.df["id"].values

X = np.vstack(X)

For good measure, we switch back to R to run tSNE. The matrix X, built in Python, is accessed in the R console below via reticulate::py$X.

set.seed(999) ##

tsne <- Rtsne::Rtsne(X = as.matrix(reticulate::py$X),

check\_duplicates = FALSE)

tsne\_clean <- data.frame(reticulate::py\_to\_r(df), tsne$Y) %>%

mutate(t1 = gsub('(further|farther)', '\\<\\1\\>', text, ignore.case = T),

t2 = stringr::str\_wrap(string = t1,

width = 20,

indent = 1,

exdent = 1),

id = row\_number()) %>%

select(id, form, X1, X2, t1, t2)

The scatter plot below summarizes contextual embeddings for individual tokens of further-farther. So, a nice space for further used adjectivally on the right side of the plot. Other spaces less obviously structured, and some confused spaces as well where speakers seem to have quite a bit of leeway.

p <- ggplot2::ggplot(tsne\_clean,

aes(x = X1,

y = X2,

color = form,

text = t2,

key = id )) +

geom\_hline(yintercept = 0, color = 'gray') +

geom\_vline(xintercept = 0, color = 'gray') +

geom\_point(alpha = 0.5) +

theme\_minimal() +

ggthemes::scale\_colour\_economist() +

ggtitle('further-farther')

plotly::ggplotly(p,

tooltip = 'text') %>%

plotly::layout(autosize = T)