

Similarity Calculation

Erin M. Buchanan

9/12/2019

First, words will be selected from the SUBTLEX projects that are:

- 1) Not stopwords (the, an, of)
- 2) Three or more characters
- 3) In the top 10,000 words

We are using the Lg10WF data - this is the log of the word frequency from the subtitle counts. Using log of frequency is advantageous, we can compare the frequencies across datasets, as well as deal with the large skew present in frequency data.

```
##Library to read excel files
library(readxl)

##Library for stopwords
library(stopwords)

library(dplyr)

##Import the US English Data
US_freq <- read_excel("similarity_data/SUBTLEXusfrequencyabove1.xls")
##Import the Dutch Data
load("similarity_data/SUBTLEX-NL.cd-above2.Rdata")
NL_freq <- subtexl.nl.cdgt2
rm(subtexl.nl.cdgt2)

##lower case all words
US_freq$Word <- tolower(US_freq$Word)
NL_freq$Word <- tolower(NL_freq$Word)

##Grab the top 10,000 words
US_subset <- US_freq %>% #data frame
  filter(!Word %in% stopwords(language = "en")) %>% #take out stop words
  filter(nchar(Word) >= 3) %>% #words greater than or equal 3
  arrange(Lg10WF) %>% #sort by Log10WF
  top_n(10000) #Take the top 10k

NL_subset <- NL_freq %>% #data frame
  filter(!Word %in% stopwords(language = "nl")) %>% #take out stop words
  filter(nchar(Word) >= 3) %>% #words greater than or equal 3
  arrange(Lg10WF) %>% #sort by Log10WF
  top_n(10000) #Take the top 10k

##these libraries will be combined with translate R
##this service is not free, uses google's API
##will work with the university to see if we have
##a service already for this type of task
#library(translateR)

write.csv(US_subset, "US_subset_words.csv", row.names = F)
write.csv(NL_subset, "NL_subset_words.csv", row.names = F)
```

In this next step, I worked with the online interface for snaut using the pre-defined English/Dutch lemma space. Spaces are currently only available for English and Dutch built with the SUBTLEX projects. Their paper for the snaut model details the creation of the space, and the model can be created from the raw SUBTLEX files. If accepted, we will work with the original authors (Pawel Mandera and Marc Brysbaert) to create models for all the target languages. The SUBTLEX data is partially online in raw form (i.e., complete movie subtitles, rather than summary statistics), but the lines of the data are currently scrambled for copyright purposes. We will also use a full word semantic space - this semantic space for example purposes is only lemmas (i.e., root word forms, running -> run).

If SUBTLEX data cannot be made available, we will use semantic feature production norms or free association norms to create highly similar pairs across languages and translate into languages that do not have feature production or free association norms available.

```
knitr::include_graphics("snuat_example.PNG")
```

English, lemmas, CBOW, 300 dimensions, window 6, UKWAC + subtitle corpus

[More info](#)[★ Neighbours](#)[Matrix](#)[Pairwise](#)[+ Analogy](#)

Words/phrases:

stating
washes
accordingly
exposing
dipped

[Calculate](#)

Neighbours:

Distance:

[Load from a file](#)

Here's an example of the Dutch data - I am processing the information produced from snaut into a file with the top five cosines for each prime word that is included in the semantic space. Cosine is a similarity value that indicates 0 (no similarity) to 1 (perfect similarity), and can be interpreted in the same fashion as a correlation.

```
library(readxl)

##read in the excel file from snaut
NL_snaut <- read_excel("NL_snaut.xlsx", col_names = FALSE)

## New names:
## * `` -> ...1
## * `` -> ...2
##give real column names
colnames(NL_snaut) <- c("Prime", "cosine")

##figure out the priming words because the formatting is bad
##imported as small 6 line tables with multiple headers
target_words <- subset(NL_snaut,
                       cosine == "Distance")

##put the prime word with it's targets
NL_snaut$Target <- rep(target_words$Prime, each = 7)

##take out all the extra headers and repeated prime words
NL_snaut <- subset(NL_snaut,
                  cosine != "Distance" & cosine > 0)

##print out the final list if you have the right encoding
head(NL_snaut, n = 15)
```

```
## # A tibble: 15 x 3
##   Prime      cosine Target
##   <chr>    <chr>   <chr>
## 1 betalen    0.38  aanbetaling
## 2 renteloos  0.4   aanbetaling
## 3 geld       0.41  aanbetaling
## 4 dollar     0.42  aanbetaling
## 5 contanten  0.43  aanbetaling
```

```
## 6 koopje          0.6  aanbevelen
## 7 cheddar         0.61 aanbevelen
## 8 langskomen      0.61 aanbevelen
## 9 waarderen       0.62 aanbevelen
## 10 natuurvoedingswinkel 0.62 aanbevelen
## 11 visitatiecommissie 0.61 aanbeveling
## 12 kabinetsvoorstel 0.63 aanbeveling
## 13 openbaarmaking 0.63 aanbeveling
## 14 advies         0.64 aanbeveling
## 15 aanvinken      0.64 aanbeveling
```

Here's an example of the US data:

```
##read in the excel file from snaut
US_snaut <- read_excel("US_snaut.xlsx", col_names = FALSE)
```

```
## New names:
## * `` -> ...1
## * `` -> ...2

##give real column names
colnames(US_snaut) <- c("Prime", "cosine")

##figure out the priming words because the formatting is bad
##imported as small 6 line tables with multiple headers
target_words <- subset(US_snaut,
  cosine == "Distance")

##put the prime word with it's targets
US_snaut$Target <- rep(target_words$Prime, each = 7)

##take out all the extra headers and repeated prime words
US_snaut <- subset(US_snaut,
  cosine != "Distance" & cosine > 0)

##print out the final list
head(US_snaut, n = 15)
```

```
## # A tibble: 15 x 3
##   Prime      cosine Target
##   <chr>    <chr> <chr>
## 1 flee    0.49  abandon
## 2 abdicate 0.51  abandon
## 3 abandoned 0.51  abandon
## 4 return  0.52  abandon
## 5 leave   0.52  abandon
## 6 deserted 0.39  abandoned
## 7 abandon  0.51  abandoned
## 8 orphaned 0.52  abandoned
## 9 found    0.54  abandoned
## 10 disused 0.54  abandoned
## 11 abdominal 0.35  abdomen
## 12 umbilicus 0.42  abdomen
## 13 duodenum 0.42  abdomen
## 14 chest    0.43  abdomen
## 15 mediastinum 0.43  abdomen
```

##note the NA values here are expected, other columns have values but there's 129 of them, which is silly to print out

The data here would then be merged across languages (not shown because of cost - see above). Native speakers will be recruited to ensure the accurate translation of word pairs. The related word pairs ($n = 1000$) will be selected from the list using each concept only once, favoring pairs with translations in most languages. If a selected pair does not exist in a language, translation from a Native speaker will be used to create that pair. Words will also be cross referenced for polysemy (i.e., multiple meanings) and these will be restricted when possible.

Next, these words would be combined together with the original frequency data and the other possible variables available mentioned in the proposal like imageability, age of acquisition, etc. LexOPS contains many of these variables already in English, but the sources for the non-English languages will be pulled from the LAB database (www.wordnorms.com; Buchanan et al., 2019). These will be compiled for inclusion in the R package for the project.

```
library(LexOPS)
##load the pretty data Jack has made
data("lexops")

##code example of data merging
```

```

##in reality, we would separate these databases and merge one at a time
##mostly because they would not be pre-arranged in non-English languages
US_stimuli <- merge(US_snaut, lexops, by.x = "Target", by.y = "string", all.x = T)

##make the columns indicate that's for targets
colnames(US_stimuli)[4:66] <- paste("T_", colnames(US_stimuli)[4:66], sep = "")

##merge again for prime
US_stimuli <- merge(US_stimuli, lexops, by.x = "Target", by.y = "string", all.x = T)

##make columns clear it's for the prime
colnames(US_stimuli)[67:129] <- paste("P_", colnames(US_stimuli)[67:129], sep = "")

head(US_stimuli[ , 1:5])

##      Target      Prime cosine T_CMU.1letter T_CMU.PrN
## 1  abandon      flee  0.49          <NA>      NA
## 2  abandon abdicate  0.51          <NA>      NA
## 3  abandon abandoned 0.51          <NA>      NA
## 4  abandon  return  0.52          <NA>      NA
## 5  abandon  leave  0.52          <NA>      NA
## 6 abandoned deserted 0.39          <NA>      NA

```

Unrelated pairs are created by rearranging the presentation of related pairs, ensuring that the pairs are not related in the original semantic space. This process will be a random assignment of primes to targets, and then cross checked in their respective semantic spaces. Note that unrelated pairs are only randomized once - so that each unrelated combination only occurs once (i.e., TREE-NURSE is the unrelated pair for the NURST target; while DOCTOR-NURSE is the related pair for the NURSE target). The Wuggy program will be used to create nonwords from the final list (Keuleers & Brysbaert, 2010).