Estimations of Quantitative Measures

Chris Bentz

05/01/2023

Load libraries

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2"). For the Hrate package this is different, since it comes from github. The devtools library needs to be installed, and then the install_github() function is used.

```
library(stringr)
library(ggplot2)
library(ggrepel)
library(plyr)
library(entropy)
library(ggExtra)
library(gsubfn)
```

```
## Loading required package: proto
```

```
#library(devtools)
#install_github("dimalik/Hrate")
library(Hrate)
```

List files

Create list with all the files in the directory "data".

```
# give file paths to the files to be processed
file.list <- list.files(path = "~/Github/NaLaFi/data",
                        recursive = T, full.names = T)
head(file.list)
```

```
## [1] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0001.txt"
## [2] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0002.txt"
## [3] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0003.txt"
## [4] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0004.txt"
## [5] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0005.txt"
## [6] "/home/chris/Github/NaLaFi/data/non-writing/animal/animal_bhg_0006.txt"
```

```
length(file.list)
```

```
## [1] 239
```

```
#same for the teddi sample (downloaded from https://drive.switch.ch/index.php/s/MJv7xFkzqlzFnOy)
file.list.teddi <- list.files(path = "~/Data/TeDDi_dumps/Teddi_unifiedformat",
                        recursive = T, full.names = T)
head(file.list.teddi)
```

```
## [1] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/abk_pro_1.txt"
## [2] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/aey_nfi_1.txt"
## [3] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/amp_nfi_1.txt"
## [4] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/ape_nfi_1.txt"
## [5] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/apu_nfi_1.txt"
## [6] "/home/chris/Data/TeDDi_dumps/Teddi_unifiedformat/arn_nfi_1.txt"
```

```
length(file.list.teddi)
```

```
## [1] 23326
```

```
# downsample the number of teddi files
file.list.teddi <- sample(file.list.teddi, 1000)

#concatenate the two lists
file.list.combined <- c(file.list, file.list.teddi)
length(file.list.combined)
```

```
## [1] 1239
```

Estimations per file

```
# set counter
counter = 0
# set the maximal number of units (n) to be used for analysis
n = 1000
# initialize dataframe to append results to
estimations.df <- data.frame(filename = character(0), corpus = character(0),
                             subcorpus = character(0), huni.chars = numeric (0),
                             hrate.chars = numeric(0), ttr.chars = numeric(0),
                             rm.chars = numeric(0))

# start time
start_time <- Sys.time()
for (file in file.list.combined)
{
  try({ # if the processing failes for a certain file, there will be no output for this file,
  # but the try() function allows the loop to keep running

  # basic processing
  # loading textfile
  textfile <- scan(file, what = "char", quote = "", comment.char = "",
                   encoding = "UTF-8", sep = "\n" , skip = 0)
  # remove the header lines beginning with '#'
  textfile <- textfile[!grepl('^#.*$', textfile)]
  # remove annotations marked by '<>'
  textfile <- gsub("<.*>","",textfile)
  # print(head(textfile))
  # get filename
  filename <- basename(file)
  # print(filename) # for visual inspection
  # get corpus category
  if (grepl('non-writing', file)){
```

```
  corpus <- 'non-writing'
} else {
  corpus <- 'writing'
}
# get subcorpus category
if (grepl('Teddi', file)){
  subcorpus <- 'teddi'
} else {
  subcorpus <- sub("_.*", "", filename)
}
# print(subcorpus) # for visual inspection

# Split into individual characters/signs
# remove tabs and parentheses, as well as star signs `*' and plus signs `+´
# note that this might have to be tuned according to the text files included
textfile <- str_replace_all(textfile, c("\\\t" = "", "\\(" = "", "\\)" = "",
                                        "\\]" = "", "\\[" = "",  "\\}" = "",
                                        "\\{" = "", "\\*" = "", "\\+" = ""))
# split the textfile into individual utf-8 characters. Note that white spaces are
# counted as utf-8 characters here.
chars <- unlist(strsplit(textfile, ""))
chars <- chars[1:n] # use only maximally n units
chars <- chars[!is.na(chars)] # remove NAs for vectors which are already shorter than n
# chars <- chars[chars != " "] # remove white spaces from character vector

# unigram entropy estimation
# calculate unigram entropy for characters
chars.df <- as.data.frame(table(chars))
# print(chars.df)
huni.chars <- entropy(chars.df$Freq, method = "ML", unit = "log2")

# entropy rate estimation
# the values chosen for max.length and every.word will crucially
# impact processing time. In case of "max.length = NULL" the length is n units
# here use the try() function since entropy rate estimation can fail when particular
# special characters are in the strings
hrate.chars <- try(get.estimate(text = chars, every.word = 1, max.length = NULL))
if (class(hrate.chars) == "numeric") {
  hrate.chars <- hrate.chars
} else {
  hrate.chars = "NA"
}

# calculate type-token ratio (ttr)
ttr.chars <- nrow(chars.df)/sum(chars.df$Freq)

# calculate repetition measure according to Sproat (2014)
# the overall number of repetitions is the sum of frequency counts minus 1.
R <- sum(chars.df$Freq-1)
# calculate the number of adjacent repetitions
r = 0
if (length(chars) > 1){
  for (i in 1:(length(chars)-1)){
```

```
      if (chars[i] == chars[i+1]){
        r = r + 1
      } else {
        r = r + 0
      }
    }
    # calculate the repetition measure
    rm.chars <- r/R
  } else {
    rm.chars <- "NA"
  }

  # append results to dataframe
  local.df <- data.frame(filename, corpus, subcorpus, huni.chars,
                         hrate.chars, ttr.chars, rm.chars)
  estimations.df <- rbind(estimations.df, local.df)
  # counter
  counter <- counter + 1
  # print(counter)
  })
}
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 3.095584 mins
```

```
#estimations.df
```

Safe outputs to file

```
write.csv(estimations.df, paste("~/Github/NaLaFi/results/estimation",
                                paste(n, "chars.csv", sep = ""), sep = ""), row.names = F)
```