Machine Learning for Psychologists

Gentle Introduction to NLProc in R w/ {Quanteda} & {Text}

Contents

Preprocessing

The **Quanteda** Package and offshoots are heavy-lifters for all text analyses in R.

```
#Enter Quanteda
library(tidyverse)
library(quanteda)

#Tokenizing
string <- "Computational analysis of linguistic data, requires translating texts into quantifiable meas
toks <- string %>% tokens()
toks_no_comma <- string %>% tokens(remove_punct = T)
#Word stems
stems <- toks %>% tokens_wordstem()
#Lemmatizing
lemmas <- string %>% textstem::lemmatize_strings() %>% quanteda::tokens()

#some sample text - can read about it more by using ?quanteda::data_char_ukimmig2010
uk_text <- quanteda::data_char_ukimmig2010
#sort as data.frame
uk_txt_df <- data.frame(text=uk_text)

#tokenize the texts
uk_toks <- tokens(uk_txt_df$text, remove_punct = T) %>%
  tokens_tolower()
```

```r
#create a document-feature matrix (dfm)
#more about the function, and the variables used at: ?dfm
uk_dfm <- dfm(uk_toks) %>%
  dfm_remove(stopwords('en')) %>%
  dfm_wordstem()

#create frequency table
doc_freq <- docfreq(uk_dfm)
#remove anything of freq <2
dim(uk_dfm)
```

```
## [1]    9 1236
```

```r
uk_dfm <- uk_dfm[, doc_freq>=2]
dim(uk_dfm)
```

```
## [1]   9 396
```

```r
#calculate feature proportions
uk_dfm_prop <- dfm_weight(uk_dfm, scheme = c("prop"))


#turn into a data-frame
dfm_mat <- convert(uk_dfm, to = "data.frame")
```

Analysis

Once we have our tokens & DFM we can use them to: (1) Fit a Bag-of-Words (BoW) model (2) Visualize (3) Extract Keyness Scores (4) Analyze using a dictionary etc.

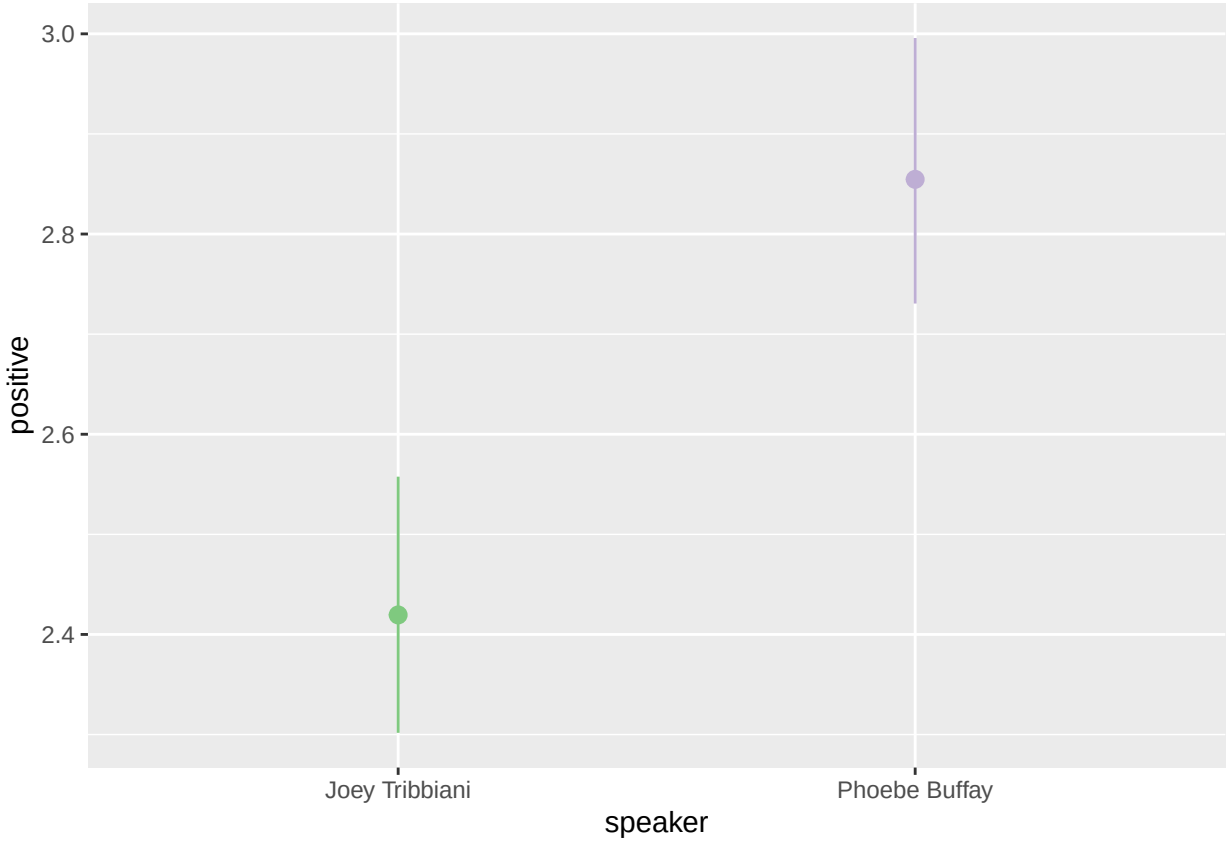
Dictionaries

We'll see how we can assess texts with 2 types of dictionaries: (a) "simple" counts - counting the proportion of words belonging to a specific category (such as positive/negative emotion)

(b) "weighted" counts (/norms) - different words carry a different weight according to their norms (e.g., "banana" is more concrete than "hope")

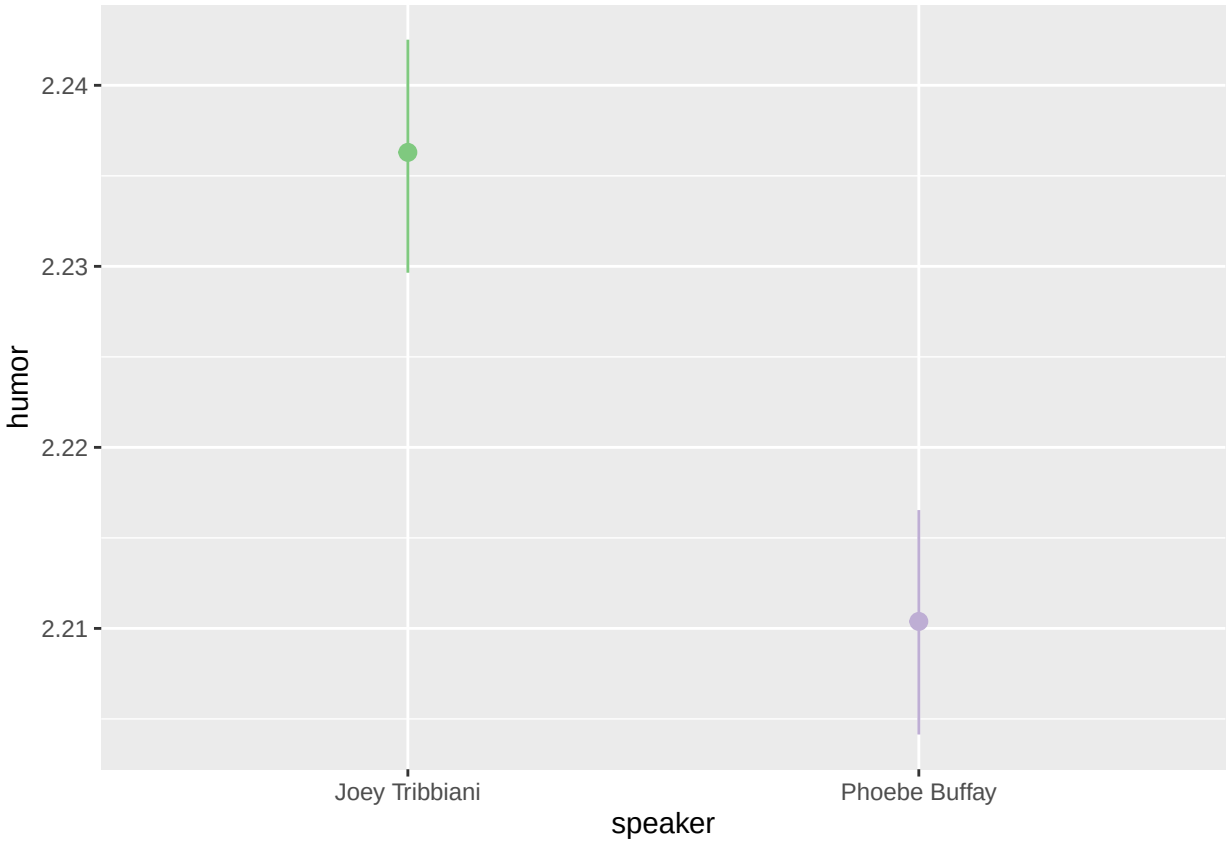
```r
#let's load some more interesting data from tidytuesday
friends <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020

#let's see who has more positive sentiment Phoebe or Joey?
pheb_joe <- friends %>%
  filter(speaker%in%c('Phoebe Buffay','Joey Tribbiani'))
pheb_joe_toks <- tokens(pheb_joe$text,remove_punct = T) %>%
  tokens_tolower() %>% tokens_remove(stopwords('en'))

#We'll use the quanteda.dictionary and the liwcalike function to test this
#along with the NRC emotion lexicon
```

```
#devtools::install_github("kbenoit/quanteda.dictionaries")
library(quanteda.dictionaries)

sentimentz <- liwcalike(pheb_joe$text,quanteda.dictionaries::data_dictionary_NRC)
sentimentz$speaker <- pheb_joe$speaker

sentimentz %>%
  ggplot(aes(x=speaker,y=positive,color=speaker))+
  stat_summary(fun.data = mean_cl_boot,show.legend = F)+
  scale_color_brewer(palette='Accent')
```




```
#now let's see about sense of humor...?
#for this we;ll use a humor-norms dictionary
humor_data <- read_csv('https://raw.githubusercontent.com/tomasengelthaler/HumorNorms/master/humor_datas
#In this dictionary words aren't only assigned to a category (e.g., sad, positive)
#but have weights of their humor norms

#a handy little function for looking up the norm-scores of words in each text
#and then averaging over the scores in each text
norm_dict <- function(now_toks,dict,word_col='word',norm_col='mean'){
  my_dict <- data.frame(word=dict[[word_col]],sentiment=dict[[norm_col]]) %>%
    quanteda::as.dictionary()
  looked_up <- quanteda::tokens_lookup(now_toks,my_dict)
  all_m_count <- looked_up %>%
      lapply(as.numeric) %>%
```

```
      sapply(mean,na.rm=T) %>%
      unname()
  return(all_m_count)
}


pheb_joe$humor <- norm_dict(pheb_joe_toks,humor_data)

#so.. who's funnier?
pheb_joe %>%
  ggplot(aes(x=speaker,y=humor,color=speaker))+
  stat_summary(fun.data = mean_cl_boot,show.legend = F)+
  scale_color_brewer(palette='Accent')
```

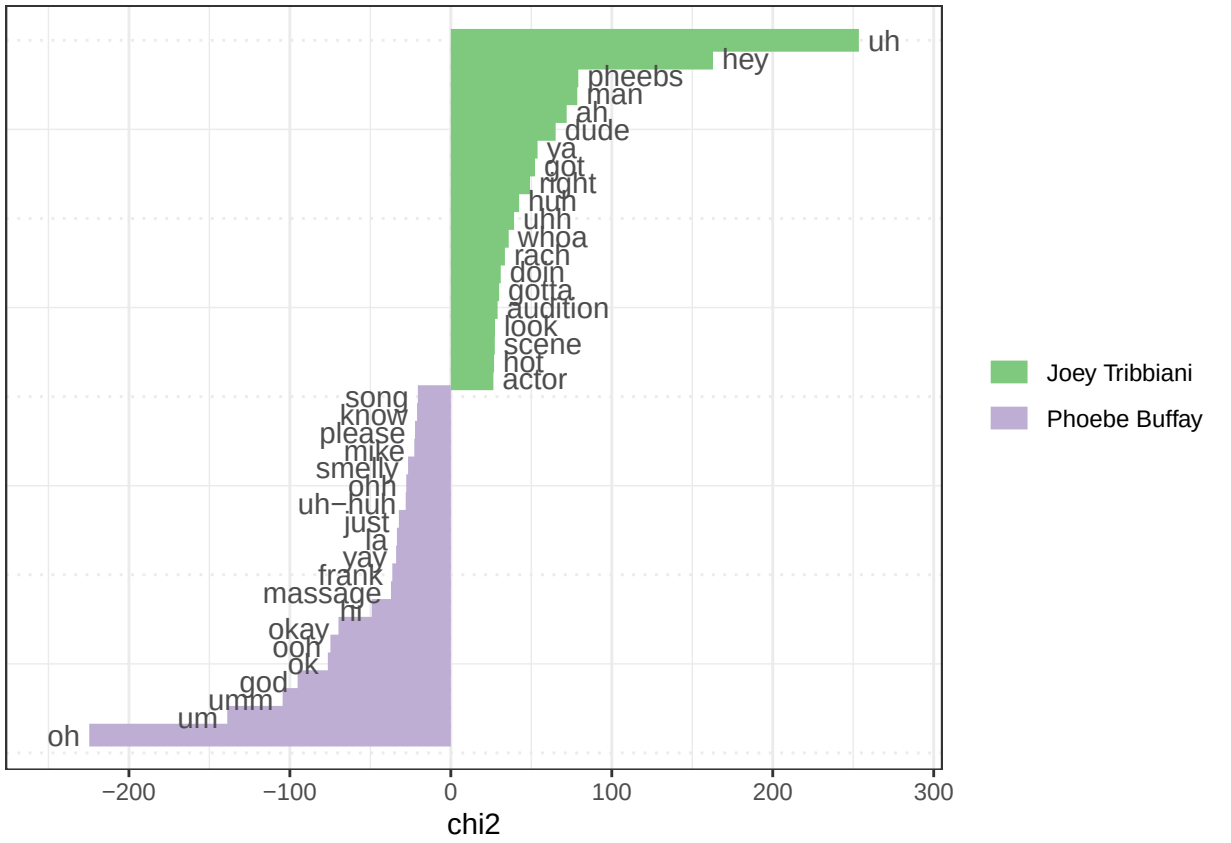


Keyness

Keyness is a metric* to describe "keywords", words that are indicative of one group but not the other. * There are actually several methods to calculate keyness score (see more at: `?quanteda.textstats::textstat_keyness`)

```
#another 2 off-shoots of the quanteda-verse which let us calculate and then plot
#the keyness scores for each speaker

library(quanteda.textstats)
library(quanteda.textplots)
```

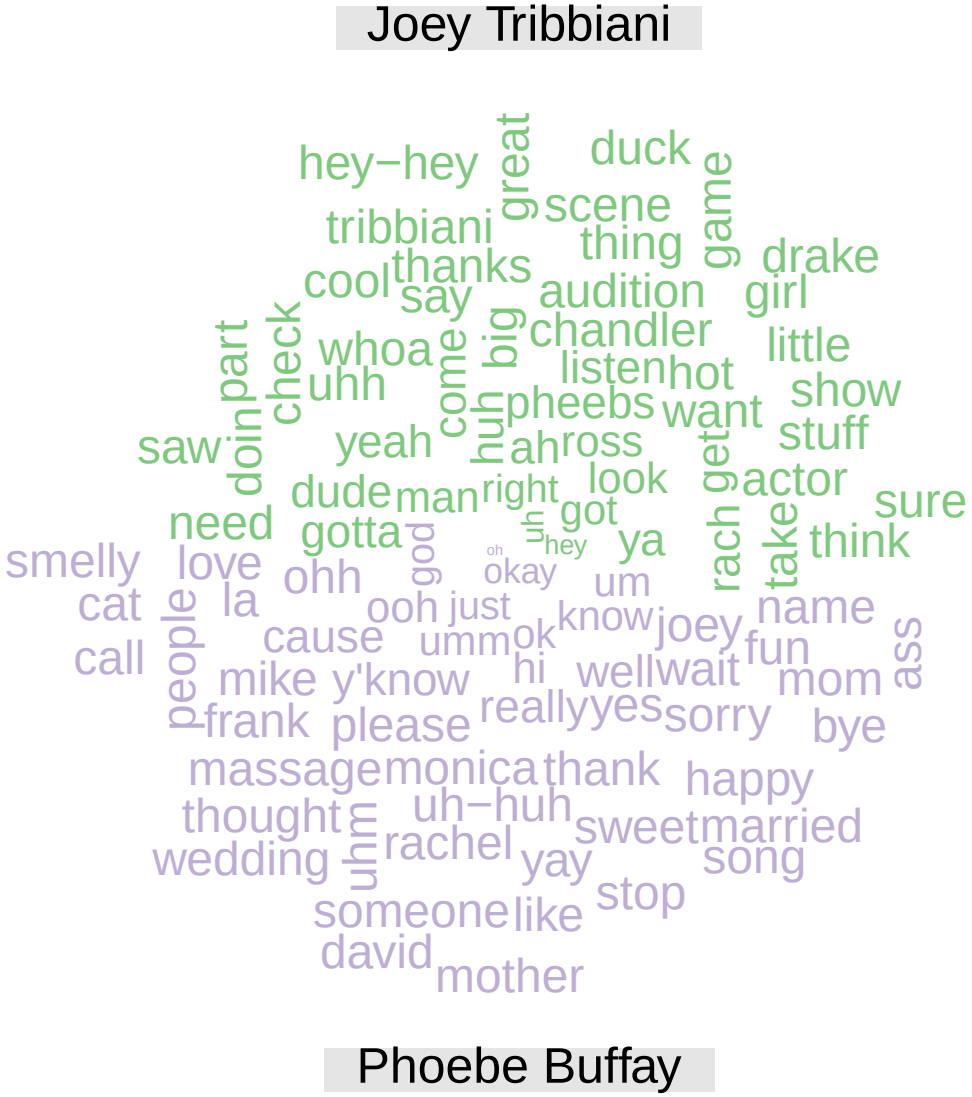
```
pheb_joe_dfm <- dfm(pheb_joe_toks) %>% dfm_group(pheb_joe$speaker)
friends_key <- textstat_keyness(pheb_joe_dfm)
textplot_keyness(friends_key,color = c("#7FC97F","#BEAED4"))
```



Word Clouds

Another way to visualize texts or compare between groups of texts are word clouds. Word clouds are often scaled according to their prevalence with more frequent words drawn larger.

```
#let's make a comparison cloud
library(wordcloud)
friends_freq_df2 <- convert(t(pheb_joe_dfm),to='data.frame')
rownames(friends_freq_df2) <- friends_freq_df2$doc_id

#plot the comparison cloud
#dev.new(width = 800, height = 800, unit = "px") #we might need it
friends_freq_df2[,2:3] %>%
  comparison.cloud(colors = c("#7FC97F","#BEAED4"),
                   max.words = 100, title.size = 2,scale=c(0.6,2))
```



Embeddings

Static embeddings - fastText

We don't have to train embeddings from scratch each time. the **`fastTextR`** pkg is an R interface with the fastText library,which has pre-trained embedding models (with models in various languages, including hebrew!). The models can be found at https://fasttext.cc/docs/en/crawl-vectors.html

```
#library(fastTextR)
#load a model
#model <- ft_load(file = 'cc.en.300.bin')
#model <- ft_load(file = file.choose())
```

```
#find nearest words (according to cosine similarity) to a given word
#ft_nearest_neighbors(model,'fast',20)

#notice there are words with identical stems (e.g. 'fast-')
#and they tend to show up as closest, so some cleaning up is sometimes needed
#wrds <- ft_nearest_neighbors(model,'fast',20)
#wrds
#wrds <- wrds[!grepl('\\bfast',names(wrds),ignore.case = T)]
#wrds <- wrds[!grepl('fast\\b',names(wrds),ignore.case = T)]
#wrds

#extract embedding vectors
#fast_ft <- ft_word_vectors(model,words = c('fast'))
#emb_ft <- ft_word_vectors(model,words = names(wrds))
#dim(emb_ft)

#cosine similarity
#quanteda.textstats::textstat_simil(x = as.dfm(emb_ft), y = as.dfm(fast_ft),
#                                   method = "cosine")


#Finally, a neat trick - analogies.
#In fasttextR, you can provide a triplet of words, and it will
#provide it's best guess for a fourth words to complete the analogy.
#the example from the pkg documentation
#ft_analogies(model,c('berlin','germany','france'))
#Nice! Let's try it out with the analogy, man-woman king-x
#ft_analogies(model,c('man','woman','king'),15)
#not so good, but queen is in there somewhere...
```

Contextual embeddings - transformers

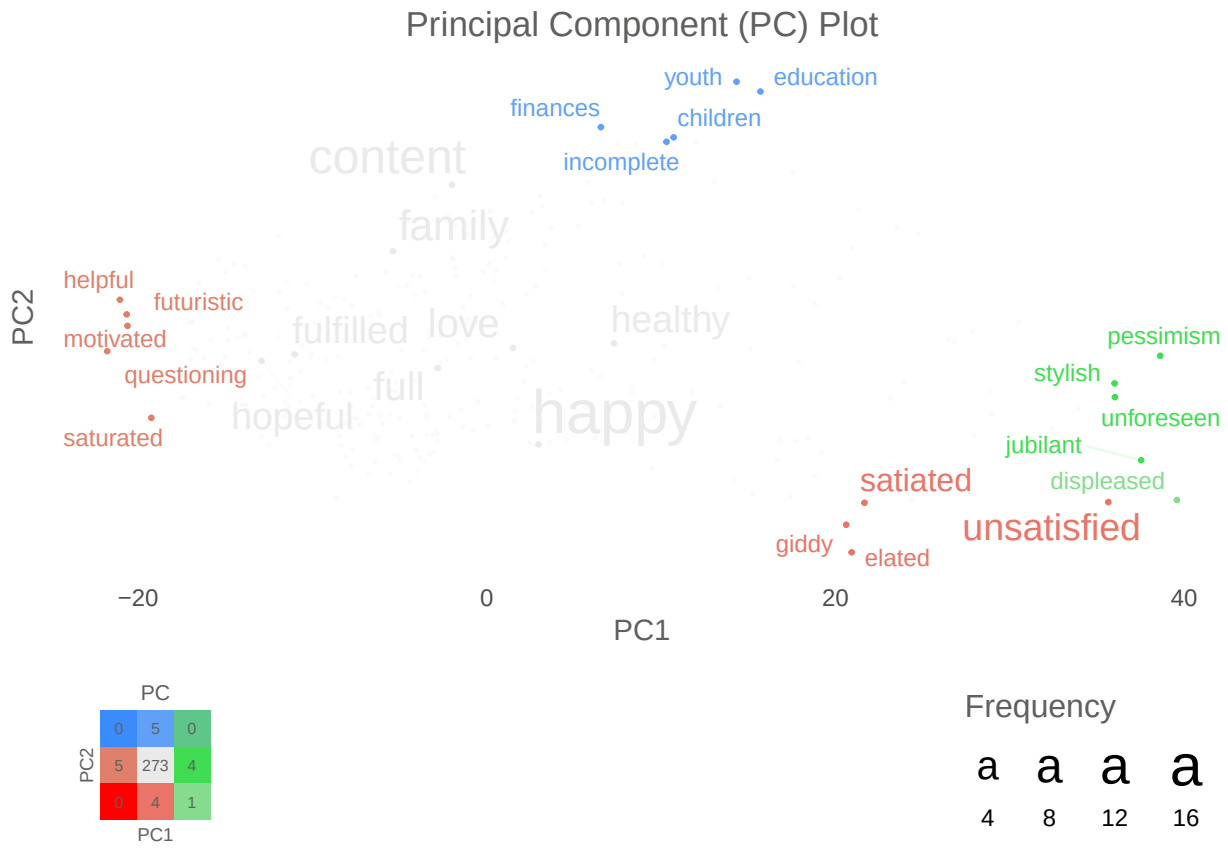
The **text** package is a convenient wrapper to run transformer models in R. You can use the package to extract static (i.e., word-) embeddings as well as contextual embeddings, basically from any package in the huggingface repository (e.g., BERT, distilBERT, roBERTa, GPT-2 etc.)

```
library(text)
pheb_joe2 <- pheb_joe %>%
  slice_sample(n=20,by=speaker)

# embedz <- textEmbed(
#   texts = pheb_joe2$text,
#   model = "bert-base-uncased",
#   layers = -2,
#   aggregation_from_tokens_to_texts = "mean",
#   aggregation_from_tokens_to_word_types = "mean",
#   keep_token_embeddings = T)

embedz <- readRDS('embedz.rds')

#The embeddings can now be used for modelling, etc.
```

```
#text similarity
textSimilarity(embedz$texts$texts[1,],embedz$texts$texts[2,])
```

```
## [1] 0.6255263
```

```
# Visualizing embeddings by plotting their first 2 Principal Components:
# (Example from pkg)
PC_projection_plot <- textPCAPlot(PC_projections_satisfactionwords_40)
PC_projection_plot$final_plot
```




```
#See text documentation for more functions: fine-tuning classification & regression models on embedding
#As well as language tasks: text generation, Q&A, summarization and more.
```