

# Assignment 6: Emoji space 2.0

In this assignment you will preprocess the words included in the streamed tweets and use them to improve the Emoji space.

## 1 Load/acquire data

### 1.1 Tweets

Load your data.

Again, to work well, latent semantic analysis requires a substantial amount of data. I recommend, in principle, using a dataset rich in Emojis containing 100,000 tweets or more. However, including words in the analysis will result substantially slower code execution. Thus, while testing your code better use a smaller amount of tweets (1,000 - 10,000).

### 1.2 Emojis

Load the new Emoji list. Remember to remove Emoji 2283, if it creates errors.

## 2 Preprocess words

Extract words and apply a series of preprocessing steps.

### 2.1 Deconstruct tweets

Split tweets in individual words using `stri_split()` and " " as the regex pattern. Then, count the number of words per tweet using `lengths()`. Use the lengths to create index variable: `index = rep(1:numberoftweets,vectoroflengths)`. The `index` variable will later allow you to reconstruct the tweets. Next, `unlist()` the tweets to create a single vector containing all the words of all tweets and call it `words`.

### 2.2 Convert to lower

Convert words to lower case using `stri_trans_tolower()`.

### 2.3 Remove Special words

Detect words in `words` that contain one of `http`, `www`, `#`, `0123456789`, `rt`, `&amp;` using `stri_detect_regex()`. Replace identified words by `NA`.

### 2.4 Remove Stopwords

Load stopwords list from [here](#) and replace all stopwords in `words` by `NA` (using `%in%`).

### 2.4 Remove punctuation

Remove all punctuation from words using `stri_replace_all_regex()`, `[:punct:]`, and the empty string.

## 2.5 Remove short words

Determine number of characters for all words using `nchar()`. Replace all words with less than 3 characters by NA.

## 2.6 Stemming

Stem words using `wordStem()` from the `SnowballC` package.

## 2.7 Reconstruct tweets

Remove from `index` and `words` all cases where `words == "NA"` (`wordStem` changes NA in "NA"; begin with `index`). Then `split()` the words and reassemble the tweets by using `paste(collapse = " ")` into `clean_tweets`.

```
# get tweets
tweets = unique(my_stream$text)[1:5000]

# get words
words = stri_split(tweets, regex = ' ')
index = rep(1:length(words), lengths(words))
words = unlist(words)

# to lower
words = stri_trans_tolower(words)

# remove all special words
pattern = 'http|www|#|@|[0-9]|rt |&'
sel = stri_detect_regex(words, pattern)
words[sel] = NA

# get and remove stopwords
words[words %in% stopwords] = NA

# remove punct
words = stri_replace_all_regex(words, pattern = '[:punct:]', replacement = '')

# remove empty and
words[nchar(words) <= 2] = NA

# stem
words = wordStem(words)

# reconstruct
index = index[words != 'NA']
words = words[words != 'NA']
clean_tweets = sapply(split(words, index), paste, collapse=' ')
```

## 3 Identify terms

Identify word terms and combine them with Emojis.

### 3.1 Remove Emojis

Remove all Emojis from words using `stri_detect_regex()`.

### 3.2 Count and select

Count words using `table()` und `select`, e.g., the most frequent 500 words.

### 3.3 Create terms

Join Emojis and selected words to create a vector of `terms`.

```
# get emojis
unis = as.character(emoji_ids$code_point)
utf8 = as.character(emoji_ids$utf8)

# remove emojis
words = words[!stri_detect_regex(words,paste(utf8,collapse='|'))]

# count
tab = table(words)
tab = tab[order(tab,decreasing = T)]

# selected words
sel_words = names(tab)[1:500] # choose more if you like

# create elements
terms = c(utf8,sel_words)
names = c(unis,sel_words)
```

## 4 Rerun last assignment

Rerun the last assignment using `terms` and `clean_tweets`.

```
# ----- create tfidf

# count emojis
text = paste(clean_tweets, collapse = ' ')
terms_found = stri_detect_fixed(text, terms)

# remove set of emojis
sel_terms = terms[terms_found]
sel_names = names[terms_found]

# # check if tweets have emojis
term_string = paste(sel_terms,collapse='|')
contains_terms = stri_detect_regex(clean_tweets, term_string)

# limit tweets
sel_tweets = clean_tweets[contains_terms]

# define term - document matrix
n_terms = length(sel_terms)
n_tweets = length(sel_tweets)
```

```
td = matrix(nrow = n_terms, ncol = n_tweets)

# iterate over emojis and tweets and count emojis
for(j in 1:n_tweets){
  if(j %% 1000 == 0) print(round(j / length(sel_tweets),2))
  td[,j] = stri_count_fixed(sel_tweets[j],sel_terms)
}
```

```
## [1] 0.2
## [1] 0.41
## [1] 0.61
## [1] 0.81
```

```
# get number of Emojis per doc
n_term_bydoc = colSums(td)

# get number of Emojis per doc
n_doc_byterm = rowSums(td > 0)

# determine idf
idf = log(n_tweets / n_doc_byterm)

# outer mat
tmp_mat = idf %o% (1 / n_term_bydoc)

# tfidf mat
tfidf_mat = td * tmp_mat

# ----- svd

# svd using RSpecra package
svd_solution = svds(tfidf_mat,300,300,0)
```

## 5 Plot result

### 5.1 Plot term space

Redo the plot of the last assignment including both Emojis and Words.

```
# ----- cosines

# get representation
term_repr = svd_solution$u * svd_solution$d
term_names = sel_names

# get cosines
term_cos = cosine(t(term_repr))

# ----- get space

# transform to distance
term_dist = (term_cos + 1) / 2
term_dist = 1 / (term_cos +.5)
```

```

# run tsne
term_space = tsne(term_dist)

## sigma summary: Min. : 0.1956 |1st Qu. : 0.4009 |Median : 0.4541 |Mean : 0.5512 |3rd Qu. : 0.633 |Max
## Epoch: Iteration #100 error is: 20.1743135156174
## Epoch: Iteration #200 error is: 1.23405123761824
## Epoch: Iteration #300 error is: 1.17848066102668
## Epoch: Iteration #400 error is: 1.1624489105796
## Epoch: Iteration #500 error is: 1.15366247867473
## Epoch: Iteration #600 error is: 1.15033010663005
## Epoch: Iteration #700 error is: 1.14905145766008
## Epoch: Iteration #800 error is: 1.14812786608726
## Epoch: Iteration #900 error is: 1.14792349548172
## Epoch: Iteration #1000 error is: 1.14780293583931

# ----- add emoji

# add emojis
add_emoji = function(filename, x, y, cex, match = T, jitter = 2){
  pic = readPNG(filename)
  dims = dim(pic)[1:2]
  usr = par()$usr
  dix = diff(usr[1:2])
  diy = diff(usr[3:4])
  if(match == T) ar = diy / dix else ar = 1
  x_jitter = jitter*runif(1,-dix*.01,+dix*.01)
  y_jitter = jitter*runif(1,-diy*.01,+diy*.01)
  rasterImage(pic,
    x-cex/2 + x_jitter,
    y-(ar*cex/2) + y_jitter,
    x+cex/2 + x_jitter,
    y+(ar*cex/2) + y_jitter,
    interpolate=TRUE)
}

# ----- term space

# plot
plot.new();par(mar=c(0,0,0,0));plot.window(xlim = range(term_space[,1]),ylim = range(term_space[,2]))

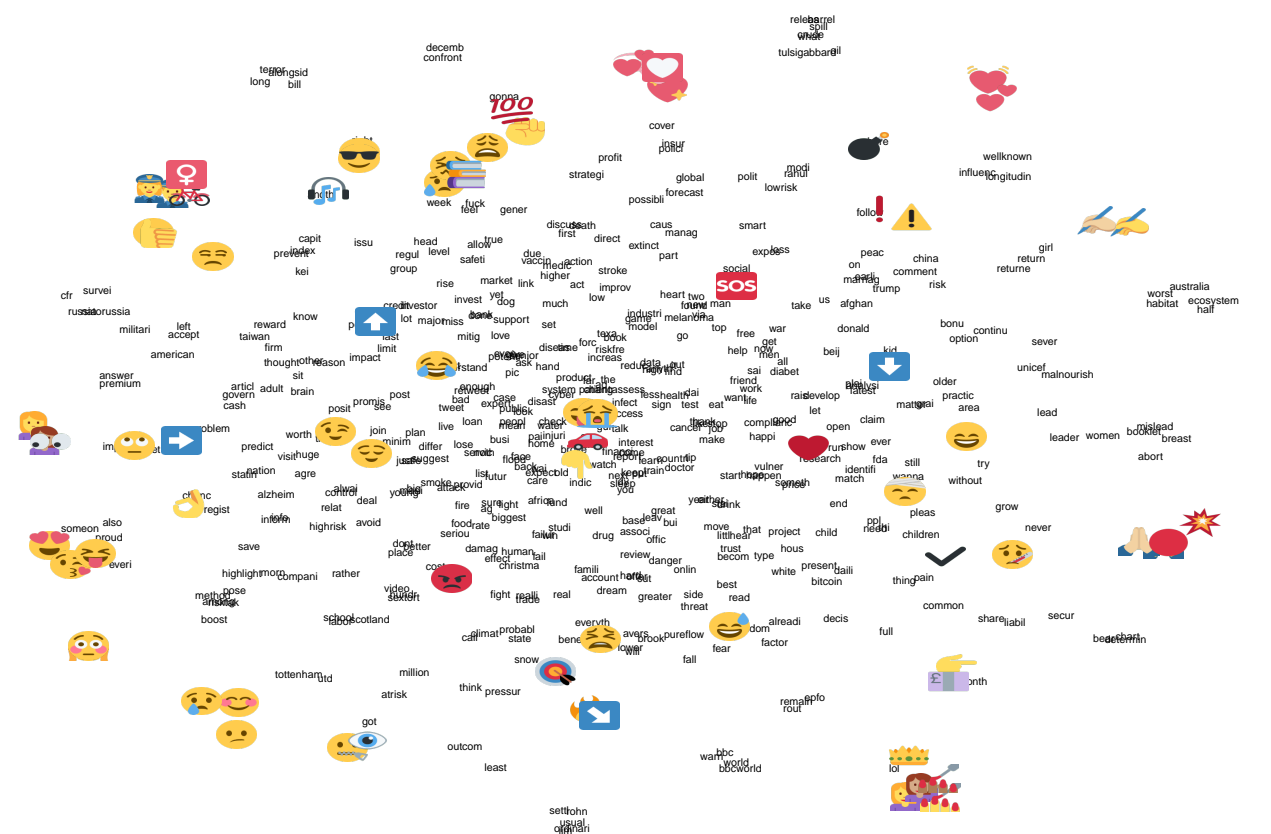
# plot words
word_ind = which(!grepl('U+',sel_names))
for(i in 1:nrow(term_space)){
  text(term_space[word_ind[i],1]+runif(1,-2,2),
    term_space[word_ind[i],2]+runif(1,-2,2),
    labels = sel_names[word_ind[i]],cex=.3,font=1)
}

```

```

# plot emojis
emoji_ind = which(grepl('U+',sel_names))
for(i in 1:nrow(term_space)){
  path = paste0(DIR,'emoji_imgs/',sel_names[emoji_ind[i]],'.png')
  if(file.exists(path)){
    add_emoji(path,term_space[emoji_ind[i],1],term_space[emoji_ind[i],2],3,match = T)
  } else {
    #add_emoji(paste0(DIR,'NAicon.png'),term_space[i,1],term_space[i,2],20,match = T)
  }
}

```



## 5.2 Plot closest associates

Plot the  $n$  most frequent Emojis and their  $k$  closest associates (in terms of cosine similarity). Place Emojis in the  $n$  rows of the left-most column and then fill the columns of each row from left to right with the ' $k$ ' closest associates. Choose  $k$  and  $n$  somewhere between 10 and 30. Add Emojis using the `add_emoji()` and the associates using `text()`.

```

# ----- closest associates

# count terms
text = paste(clean_tweets, collapse = ' ')
cnts = stri_count_fixed(text, sel_terms)

# determine indices of Emojis and words
emj = which(grepl('U+',sel_names))
wrds = which(!grepl('U+',sel_names))

```














```

# determine frequent emojis
frq_emj = which(cnts[emj] > 1)

mat = matrix(ncol=10, nrow=length(frq_emj))
for(i in 1:length(frq_emj)){
  cos = term_cos[frq_emj[i], wrd]
  ord = order(cos,decreasing=T)
  mat[i,] = sel_names[wrds][ord[1:10]]
}
mat = mat[order(cnts[frq_emj],decreasing = T),]
frq_emj = frq_emj[order(cnts[frq_emj],decreasing = T)]

# plot
plot.new();par(mar=c(0,0,0,0));plot.window(xlim = c(-.5,10.5), ylim = c(.5,13.5))
for(i in 1:13) {
  path = paste0(DIR,'emoji_imgs/',sel_names[frq_emj[i]],'.png')
  add_emoji(path,0,14-i,1,match = T,jitter=0)
  text(1:10,rep(14-i,10),labels = mat[i,])
}

```

 gonna possibli death think find wai returne best survei on  
 return risk free reduc bet ad man war the ag  
 extra month trial free decemb share read pressur atrisk pic  
 tweet retweet follow back ppl low plantulsigabbardthi spill  
 extra month trial free read share biggestpressurdecemb atrisk  
 tweet retweet follow back low ppl plan thitulsigabbardspill  
 offer daili cut join extra indic leav morn industri involv  
 proud someon everi also game keep market you week work  
 pain love thing try feel anoth someone realli pai real  
 lot decemb part sure fight huge sever industri action reason  
 gonna even highriskamerican go grow global practic trust prevent  
 leav everi breast groupPamericamilitari onlin week side share  
 list move servic seriou global leader pose action suggest grow

### 5.3 Post your results

Post your results on twitter

END