

# Assignment 7: Emoji space 3.0

In this assignment you will do rerun the analyses of the past two assignments using `word2vec`, a neural network developed by Google to learn associations between terms.

## 1 Process tweets

Rerun assignment 6 up to paragraph 2.7.

### 1.1 Split Emojis

`word2vec` must be supplied with a text file containing a single character string of all of the tweets, where all words are separated with white space ' '. Many Emojis, however, occur right after each other without any intervening characters or white space. This implies that, unless special measures are taken, many sequences of Emojis will be interpreted as a single string. To avoid this, go through the vector of cleaned words, 1) test if non-ascii characters are contained using `stri_detect_regex(word, '^[[:ascii:]]')`, 2) extract, if necessary, all Emojis using `stri_extract_all(word, regex = '^[[:ascii:]]')[[1]]`, 3) extract, if necessary, any ascii characters in the string using `stri_replace_all_regex(word, '', '^[[:ascii:]]')`, and then 4) write at the position of the original the vector of extracted Emojis. Best do this using `lapply(words, function)`, which takes a vector of words, applies a function (written by you) and returns a `list` (only a `list` allows to add vectors into a sequence of words). Then extract the sequence of words using `unlist()`.

### 1.2 Write Input File

To create the input file for `word2vec`, collapse the vector of cleaned words into a single string using `paste(,collapse='')` and write it into a file in your folder. To do this use a slightly different approach than before. First open up a connection to a (new) file using `my_file_connection = file(my_path/my_filename.txt,'wb')`. In case the file does not exist R will create it. `wb` means that the file is written in binary, which will hopefully circumvent all encoding issues. Then write the collapsed string to the file using `writeln(my_collapsed_string, my_file_connection)`. Finally, close the connection using `close(my_file_connection)`.

## 2 Word2Vec

### 2.1 Install word2vec

Different from the other packages that you have used, the `word2vec` implementation is not available from CRAN, but from GitHub. To install the package, first install the `devtools` package (from CRAN) using the regular `install.packages()` command. Then you can use `devtools`' `install_github("bmschmidt/wordVectors")` to install the `word2vec` implementation. Note that since the implementation is based on C/C++, Windows users may also have to install `Rtools`. You find the file including installation instructions here: <https://cran.r-project.org/bin/windows/Rtools/index.html>.

### 2.2 Train Model

Train your `word2vec` model using `train_word2vec`. Provide it with `my_path/my_filename.txt` as the first argument and `'my_path/my_filename.bin'` as the second. The second file will, in case it doesn't exist, be created by the function. The remaining arguments are optional. Consider playing around with `vectors`, which determines the number of latent dimensions (comparable to LSA), `window` which determines the maximum distance between words to consider them as a co-occurring word pair, `negative_samples`, which amount of learning per pair (higher number = more learning, but slower execution). Also, if your computer has multiple

(real or virtual cores) consider using `threads` to run the analysis in parallel. To find out, how many cores are available on your computer use `detectCores` from the `parallel` package. As with LSA `word2vec` will require some time to finish. I recommend first using a small subset of your data to make sure everything works correctly. Also make sure to assign the output, i.e., `my_model = train_word2vec(your_arguments)`.

### 2.3 Assess Model

There are functions to play around and probe the model. Use, for instance, `closest_to(my_model, word, n)` to view the `n` closest associates of a `word`.

## 3. Evaluate and plot

Rerun (some of) the plots of assignment 5 and 6 and post them on twitter. To calculate the cosine similarities use `cosineDist`. Possibly restrict the analysis to some of the Emojis/words.

**END**