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 `tolower`

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 stopword list from **here** and replace all stopwords in `words` by `NA` (using `%in%`).

2.4 Remove punctuation

Remove all punction 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`.

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`.

4 Rerun last assignment

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

5 Plot result

5.1 Plot term space

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

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()`.

5.3 Post your results

Post your results on twitter

END