Total Drama World Tour Songs

Total Drama World Tour Songs Analysis

As a bonus to the Caparezza's songs analysis, here is an analogue notebook which focuses on songs from *Total Drama World Tour* cartoon.

```
library(geniusr)
library(tidyverse)
```

```
## Warning: il pacchetto 'lubridate' è stato creato con R versione 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.0     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.0
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflict
s to become errors
```

```
library(tidytext)
```

```
## Warning: il pacchetto 'tidytext' è stato creato con R versione 4.2.3
```

```
library(quanteda)
```

```
## Warning: il pacchetto 'quanteda' è stato creato con R versione 4.2.3
```

```
## Package version: 3.3.1
## Unicode version: 13.0
## ICU version: 69.1
## Parallel computing: 8 of 8 threads used.
## See https://quanteda.io for tutorials and examples.
```

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

```
## Warning: il pacchetto 'quanteda.textstats' è stato creato con R versione 4.2.3
```

```
library(udpipe)
```

```
## Warning: il pacchetto 'udpipe' è stato creato con R versione 4.2.3
```

```r
library(wordcloud)
```

```
## Warning: il pacchetto 'wordcloud' è stato creato con R versione 4.2.3
```

```
## Caricamento del pacchetto richiesto: RColorBrewer
```

```r
library(textdata)
```

```
## Warning: il pacchetto 'textdata' è stato creato con R versione 4.2.3
```

```r
library(reshape2)
```

```
## Warning: il pacchetto 'reshape2' è stato creato con R versione 4.2.3
```

```
##
## Caricamento pacchetto: 'reshape2'
##
## Il seguente oggetto è mascherato da 'package:tidyr':
##
##     smiths
```

```r
library(igraph)
```

```
## Warning: il pacchetto 'igraph' è stato creato con R versione 4.2.3
```

```
##
## Caricamento pacchetto: 'igraph'
##
## I seguenti oggetti sono mascherati da 'package:lubridate':
##
##     %--%, union
##
## I seguenti oggetti sono mascherati da 'package:dplyr':
##
##     as_data_frame, groups, union
##
## I seguenti oggetti sono mascherati da 'package:purrr':
##
##     compose, simplify
##
## Il seguente oggetto è mascherato da 'package:tidyr':
##
##     crossing
##
## Il seguente oggetto è mascherato da 'package:tibble':
##
##     as_data_frame
##
## I seguenti oggetti sono mascherati da 'package:stats':
##
##     decompose, spectrum
##
## Il seguente oggetto è mascherato da 'package:base':
##
##     union
```

Get lyrics

```r
Sys.setenv(GENIUS_API_TOKEN = "_EwGMpzt9E9rqXlKxi2cVDJekd_6zYRLNpjolP4xkXp-RSxitzHNIyMeL7TT1T
ba")

artist_id <- search_artist("Total Drama World Tour")$artist_id
artist_songs <- get_artist_songs(artist_id)

songs_ids <- c()
for (song in artist_songs$content) {
  song_id <- song$id
  songs_ids <- songs_ids %>% append(song_id)
}

songs_titles <- c()
songs_lyrics <- c()
for (i in 1:length(songs_ids)) {
  cat("Getting", i, "of", length(songs_ids), "id:", songs_ids[i], "\n")
  song <- get_song(songs_ids[i])$content
  song_title <- song$title
  song_lyrics <- list(get_lyrics_id(songs_ids[i]))
  songs_titles <- songs_titles %>% append(song_title)
  songs_lyrics <- songs_lyrics %>% append(song_lyrics)
}
```

```
## Getting 1 of 31 id: 8789649
## Getting 2 of 31 id: 6109215
## Getting 3 of 31 id: 5979806
## Getting 4 of 31 id: 5779872
## Getting 5 of 31 id: 3064618
## Getting 6 of 31 id: 7610349
## Getting 7 of 31 id: 5348279
## Getting 8 of 31 id: 6801747
## Getting 9 of 31 id: 8786380
## Getting 10 of 31 id: 8897131
## Getting 11 of 31 id: 7488358
## Getting 12 of 31 id: 5832464
## Getting 13 of 31 id: 8751707
## Getting 14 of 31 id: 3064631
## Getting 15 of 31 id: 8921824
## Getting 16 of 31 id: 5348393
## Getting 17 of 31 id: 8891552
## Getting 18 of 31 id: 5630318
## Getting 19 of 31 id: 5979774
## Getting 20 of 31 id: 8898148
## Getting 21 of 31 id: 8873960
## Getting 22 of 31 id: 8917866
## Getting 23 of 31 id: 8928172
## Getting 24 of 31 id: 8918195
## Getting 25 of 31 id: 6013830
## Getting 26 of 31 id: 8921766
## Getting 27 of 31 id: 8741703
## Getting 28 of 31 id: 8897096
## Getting 29 of 31 id: 715171
## Getting 30 of 31 id: 3065755
## Getting 31 of 31 id: 8786258
```

```r
# collapse all lyrics lines into a single text
for (i in 1:length(songs_lyrics)){
  songs_lyrics[[i]] <- songs_lyrics[[i]]$line %>% paste(collapse = "\n")
}

songs_lyrics <- unlist(songs_lyrics)

songs <- data.frame(title = songs_titles, lyrics = songs_lyrics)
```

```r
songs %>% head(1)
```

```
##              title
## 1 A Chinese Lesson
##
lyrics
## 1 A little Chinese lesson, for you\nMàn man chī means "enjoy your meal."\nMàn man chī. I
t's no raw deal\nIs it roasted eel?\nMàn man chī means "bon appétit."\nMàn man chī\nWhat do w
e have to eat?\nIt's still moving its feet!\nMàn man chī. It's dinner for four\nMàn man chī.
We've got room for more\nI think I'm nearly done for\nMàn man chī. Don't get the squirts\nMàn
man chī. We'd rather eat our shirts!\nWait, stop!\nMàn man chī\n(off-key) Màn man chī-i-i\nTh
ey love to eat on The Yangtze\nMàn man chī\nMàn man... Huh?\n**both gag and vomit**\nCody's i
n first class with me and my Love-me tea!
```

```r
doc_ids <- vector()
for(i in 1:nrow(songs)){
  id <- paste("doc", toString(i), sep = "")
  doc_ids <- doc_ids %>% append(id)
}
songs <- songs %>% mutate(doc_id = doc_ids, .before = 1)
```

Text pre-processing

```r
tdwt_corpus <- corpus(songs$lyrics, docnames = songs$id)
summary(tdwt_corpus)
```

```
## Corpus consisting of 31 documents, showing 31 documents:
##
##    Text Types Tokens Sentences
##   text1    83    145        15
##   text2   115    185         5
##   text3   150    211         5
##   text4    99    165        22
##   text5    78    148        19
##   text6   123    234        21
##   text7   119    265        45
##   text8   140    271        23
##   text9   101    159        15
##  text10   104    158        22
##  text11    67     88        10
##  text12   115    202        18
##  text13   101    188         5
##  text14   103    180         7
##  text15   123    268        44
##  text16    54    122         8
##  text17    64    184        16
##  text18   126    280        26
##  text19    65    104         3
##  text20    75    124         6
##  text21   106    170        16
##  text22    80    147        16
##  text23    50     71         1
##  text24    88    177        15
##  text25    68    133         2
##  text26   126    292        21
##  text27   140    304        36
##  text28   118    223        15
##  text29    65    106         9
##  text30    80    145         9
##  text31   100    187        26
```

```
cat(as.character(tdwt_corpus[1]))
```

```
## A little Chinese lesson, for you
## Màn man chī means "enjoy your meal."
## Màn man chī. It's no raw deal
## Is it roasted eel?
## Màn man chī means "bon appétit."
## Màn man chī
## What do we have to eat?
## It's still moving its feet!
## Màn man chī. It's dinner for four
## Màn man chī. We've got room for more
## I think I'm nearly done for
## Màn man chī. Don't get the squirts
## Màn man chī. We'd rather eat our shirts!
## Wait, stop!
## Màn man chī
## (off-key) Màn man chī-i-i
## They love to eat on The Yangtze
## Màn man chī
## Màn man... Huh?
## **both gag and vomit**
## Cody's in first class with me and my Love-me tea!
```

```
corpus_tokens <- tdwt_corpus %>%
  quanteda::tokens(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE) %>%
  tokens_tolower()
```

```
txt <- sapply(corpus_tokens, FUN=function(x) paste(x, collapse = "\n"))
udpipe_download_model(language = "english-ewt", model_dir = "resources/")
```

```
## Downloading udpipe model from https://raw.githubusercontent.com/jwijffels/udpipe.models.u
d.2.5/master/inst/udpipe-ud-2.5-191206/english-ewt-ud-2.5-191206.udpipe to resources//english
-ewt-ud-2.5-191206.udpipe
```

```
##  - This model has been trained on version 2.5 of data from https://universaldependencies.o
rg
```

```
##  - The model is distributed under the CC-BY-SA-NC license: https://creativecommons.org/lic
enses/by-nc-sa/4.0
```

```
##  - Visit https://github.com/jwijffels/udpipe.models.ud.2.5 for model license details.
```

```
##  - For a list of all models and their licenses (most models you can download with this pac
kage have either a CC-BY-SA or a CC-BY-SA-NC license) read the documentation at ?udpipe_downl
oad_model. For building your own models: visit the documentation by typing vignette('udpipe-t
rain', package = 'udpipe')
```

```
## Downloading finished, model stored at 'resources//english-ewt-ud-2.5-191206.udpipe'
##      language                                  file_model
## 1 english-ewt resources//english-ewt-ud-2.5-191206.udpipe
##
url
## 1 https://raw.githubusercontent.com/jwijffels/udpipe.models.ud.2.5/master/inst/udpipe-ud-
2.5-191206/english-ewt-ud-2.5-191206.udpipe
##   download_failed download_message
## 1           FALSE               OK
```

```
lang_model <- udpipe_load_model(file = "resources/english-ewt-ud-2.5-191206.udpipe")
outL <- udpipe_annotate(lang_model, x = txt, tokenizer = "vertical", trace = TRUE) %>%
  as.data.frame()
```

```
## 2023-05-31 09:34:09 Annotating text fragment 1/31
## 2023-05-31 09:34:09 Annotating text fragment 2/31
## 2023-05-31 09:34:10 Annotating text fragment 3/31
## 2023-05-31 09:34:11 Annotating text fragment 4/31
## 2023-05-31 09:34:11 Annotating text fragment 5/31
## 2023-05-31 09:34:11 Annotating text fragment 6/31
## 2023-05-31 09:34:12 Annotating text fragment 7/31
## 2023-05-31 09:34:13 Annotating text fragment 8/31
## 2023-05-31 09:34:13 Annotating text fragment 9/31
## 2023-05-31 09:34:14 Annotating text fragment 10/31
## 2023-05-31 09:34:14 Annotating text fragment 11/31
## 2023-05-31 09:34:14 Annotating text fragment 12/31
## 2023-05-31 09:34:15 Annotating text fragment 13/31
## 2023-05-31 09:34:15 Annotating text fragment 14/31
## 2023-05-31 09:34:16 Annotating text fragment 15/31
## 2023-05-31 09:34:16 Annotating text fragment 16/31
## 2023-05-31 09:34:16 Annotating text fragment 17/31
## 2023-05-31 09:34:17 Annotating text fragment 18/31
## 2023-05-31 09:34:18 Annotating text fragment 19/31
## 2023-05-31 09:34:18 Annotating text fragment 20/31
## 2023-05-31 09:34:18 Annotating text fragment 21/31
## 2023-05-31 09:34:19 Annotating text fragment 22/31
## 2023-05-31 09:34:19 Annotating text fragment 23/31
## 2023-05-31 09:34:19 Annotating text fragment 24/31
## 2023-05-31 09:34:19 Annotating text fragment 25/31
## 2023-05-31 09:34:20 Annotating text fragment 26/31
## 2023-05-31 09:34:21 Annotating text fragment 27/31
## 2023-05-31 09:34:21 Annotating text fragment 28/31
## 2023-05-31 09:34:22 Annotating text fragment 29/31
## 2023-05-31 09:34:22 Annotating text fragment 30/31
## 2023-05-31 09:34:23 Annotating text fragment 31/31
```

```
en_stopwords <- readLines("https://raw.githubusercontent.com/stopwords-iso/stopwords-en/maste
r/stopwords-en.txt")
```

```
## Warning in
## readLines("https://raw.githubusercontent.com/stopwords-iso/stopwords-en/master/stopwords-e
n.txt"):
## riga finale incompleta in
## 'https://raw.githubusercontent.com/stopwords-iso/stopwords-en/master/stopwords-en.txt'
```

```
outL <- outL %>% filter(!(token %in% en_stopwords) & !(lemma %in% en_stopwords))
```

```
outL %>% select(doc_id, token, lemma, upos) %>% sample_n(5)
```

```
##   doc_id  token  lemma upos
## 1  doc30 sleeps  sleep VERB
## 2  doc15 burned   burn VERB
## 3  doc18   sung   sung  ADV
## 4   doc2    hot    hot  ADJ
## 5   doc3 rocket rocket NOUN
```

```
outL_reduced <- outL %>% filter(upos %in% c("NOUN", "PROPN", "ADJ", "VERB"))
```

```
# fct_inorder preserves original order of the column
lemmatized_lyrics <- outL_reduced %>% group_by(doc_id = fct_inorder(doc_id)) %>%
  summarise(lemmatized = paste(lemma, collapse = " "))
songs <- songs %>% right_join(lemmatized_lyrics, by = "doc_id")
tdwt_corpus <- songs$lemmatized %>% corpus(docnames = songs$id)
```

```
DTM <- tdwt_corpus %>% tokens() %>% dfm()

DTM
```

```
## Document-feature matrix of: 31 documents, 711 features (95.56% sparse) and 0 docvars.
##        features
## docs    chinese lesson màn chī enjoy meal raw deal roast eel
##   text1       1      1  12  10     1    1   1    1     1   1
##   text2       0      0   0   0     0    0   0    0     0   0
##   text3       0      0   0   0     0    0   0    0     0   0
##   text4       0      0   0   0     0    0   0    0     0   0
##   text5       0      0   0   0     0    0   0    0     0   0
##   text6       0      0   0   0     0    0   0    0     0   0
## [ reached max_ndoc ... 25 more documents, reached max_nfeat ... 701 more features ]
```

Lexical Analysis

```
DTM %>% dim()
```

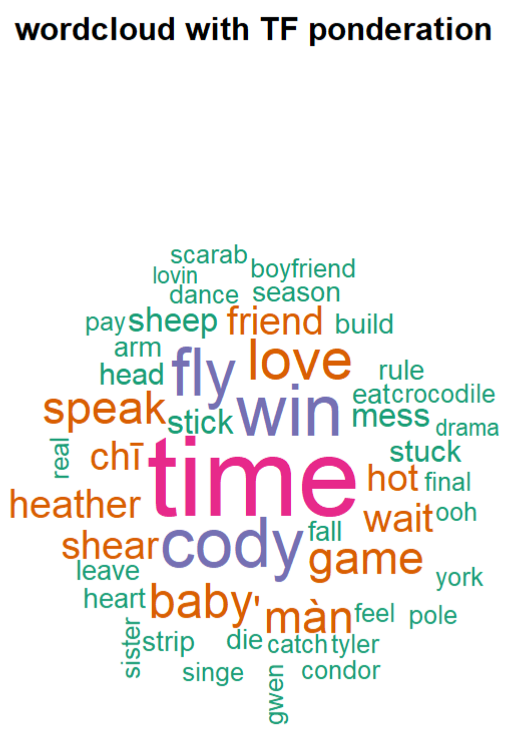
```
## [1]  31 711
```

```
words <- colnames(DTM)
freqs <- colSums(DTM)
wordlist <- data.frame(words, freqs)
wordlist %>% arrange(-freqs) %>% head()
```

```
##      words freqs
## time  time    35
## win    win    21
## cody  cody    20
## fly    fly    19
## love  love    16
## baby  baby    13
```

Data visualization

```
par(mar=c(1,1,0.5,1))
wordcloud(words = wordlist$words, freq = wordlist$freqs, scale = c(3.5, 0.35), max.words = 5
0, random.order = F,
          colors = RColorBrewer::brewer.pal(name = "Dark2", n = 4))
text(0.5, 1, "wordcloud with TF ponderation", font = 2)
```



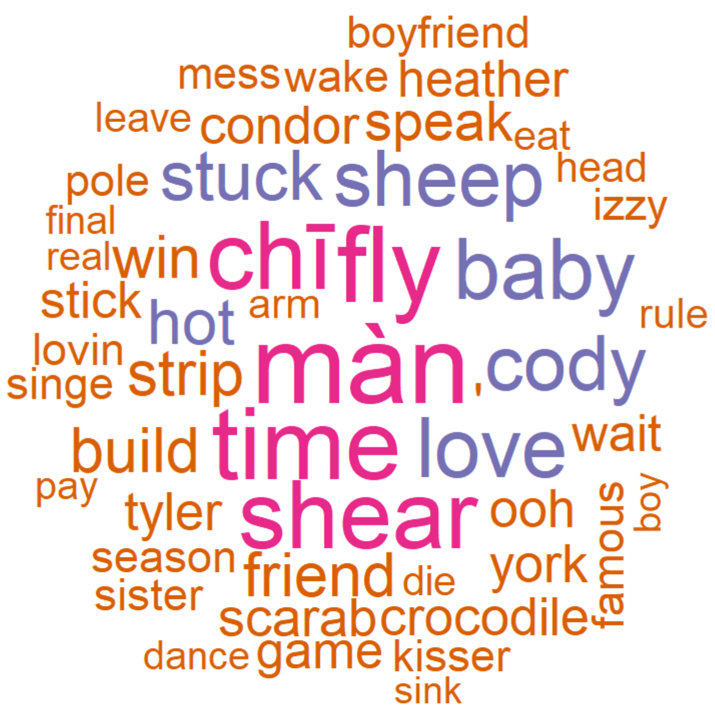
TF-IDF ponderation

```
tf_idf <- dfm_tfidf(DTM)
freqs_tf_idf <- colSums(tf_idf)
words_tf_idf <- colnames(tf_idf)
wordlist_tf_idf <- data.frame(words = words_tf_idf, freqs = freqs_tf_idf)
wordlist_tf_idf %>% arrange(-freqs) %>% head(10)
```

```
##       words    freqs
## màn     màn 17.89634
## fly     fly 16.89673
## time   time 15.74892
## chī     chī 14.91362
## shear shear 14.91362
## baby   baby 13.18513
## love   love 12.67827
## sheep sheep 11.93089
## cody   cody 11.76543
## stuck stuck 10.43953
```

```
par(mar=c(1,1,0.5,1))
wordcloud(words = wordlist_tf_idf$words, freq = wordlist_tf_idf$freqs,
          scale = c(3.5, 0.35), max.words = 50, random.order = F,
          colors = RColorBrewer::brewer.pal(name = "Dark2", n = 4))
text(0.5, 1, "wordcloud with TF-IDF ponderation", font = 2)
```

wordcloud with TF-IDF ponderation



Co-occurrence analysis

```
binDTM <- DTM %>% dfm_weight("boolean")
coocCounts <- t(binDTM) %*% binDTM
as.matrix(coocCounts[16:18, 16:18])
```

```
##         shirts wait off-key
## shirts       1    1       1
## wait         1    6       1
## off-key      1    1       1
```

```
coocTerm <- "cody"
k <- nrow(binDTM)
ki <- sum(binDTM[, coocTerm])
kj <- colSums(binDTM)
names(kj) <- colnames(binDTM)
kij <- coocCounts[coocTerm, ]

mutualInformationSig <- log(k * kij / (ki * kj))
mutualInformationSig <- mutualInformationSig[order(mutualInformationSig, decreasing = TRUE)]

dicesig <- 2 * kij / (ki + kj)
dicesig <- dicesig[order(dicesig, decreasing=TRUE)]

logsig <- 2 * ((k * log(k)) - (ki * log(ki)) - (kj * log(kj)) + (kij * log(kij))
               + (k - ki - kj + kij) * log(k - ki - kj + kij)
               + (ki - kij) * log(ki - kij) + (kj - kij) * log(kj - kij)
               - (k - ki) * log(k - ki) - (k - kj) * log(k - kj))
logsig <- logsig[order(logsig, decreasing=T)]

resultOverView <- data.frame(
  names(sort(kij, decreasing=T)[1:10]), sort(kij, decreasing=T)[1:10],
  names(mutualInformationSig[1:10]), mutualInformationSig[1:10],
  names(dicesig[1:10]), dicesig[1:10],
  names(logsig[1:10]), logsig[1:10],
  row.names = NULL)
colnames(resultOverView) <- c("Freq-terms", "Freq", "MI-terms", "MI", "Dice-Terms", "Dice",
"LL-Terms", "LL")
print(resultOverView)
```

```
##    Freq-terms Freq MI-terms       MI Dice-Terms      Dice LL-Terms       LL
## 1        cody    8  chinese 1.354546       cody 1.0000000     gwen 3.216593
## 2        wait    3   lesson 1.354546       gwen 0.4615385     feel 3.216593
## 3        time    3      màn 1.354546       feel 0.4615385   sierra 2.487852
## 4       speak    3      chī 1.354546       wait 0.4285714     hate 2.487852
## 5     heather    3    enjoy 1.354546    heather 0.4285714     wait 2.065521
## 6        gwen    3     meal 1.354546      speak 0.4000000  heather 2.065521
## 7         win    3      raw 1.354546      trust 0.4000000    speak 1.279171
## 8        feel    3    roast 1.354546      chick 0.4000000      eat 1.254096
## 9         eat    2      eel 1.354546       kick 0.4000000   friend 1.254096
## 10       love    2  appétin 1.354546      rhyme 0.4000000    stick 1.254096
```

```
coocTerm <- "boyfriend"
k <- nrow(binDTM)
ki <- sum(binDTM[, coocTerm])
kj <- colSums(binDTM)
names(kj) <- colnames(binDTM)
kij <- coocCounts[coocTerm, ]

mutualInformationSig <- log(k * kij / (ki * kj))
mutualInformationSig <- mutualInformationSig[order(mutualInformationSig, decreasing = TRUE)]

dicesig <- 2 * kij / (ki + kj)
dicesig <- dicesig[order(dicesig, decreasing=TRUE)]

logsig <- 2 * ((k * log(k)) - (ki * log(ki)) - (kj * log(kj)) + (kij * log(kij))
               + (k - ki - kj + kij) * log(k - ki - kj + kij)
               + (ki - kij) * log(ki - kij) + (kj - kij) * log(kj - kij)
               - (k - ki) * log(k - ki) - (k - kj) * log(k - kj))
logsig <- logsig[order(logsig, decreasing=T)]

resultOverView <- data.frame(
  names(sort(kij, decreasing=T)[1:10]), sort(kij, decreasing=T)[1:10],
  names(mutualInformationSig[1:10]), mutualInformationSig[1:10],
  names(dicesig[1:10]), dicesig[1:10],
  names(logsig[1:10]), logsig[1:10],
  row.names = NULL)
colnames(resultOverView) <- c("Freq-terms", "Freq", "MI-terms", "MI", "Dice-Terms", "Dice",
"LL-Terms", "LL")
print(resultOverView)
```

```
##    Freq-terms Freq  MI-terms      MI Dice-Terms      Dice  LL-Terms       LL
## 1   boyfriend    2 boyfriend 2.74084  boyfriend 1.0000000       fun 3.359166
## 2        cody    1    kisser 2.74084     kisser 0.6666667    sister 3.359166
## 3        time    1      diss 2.74084       diss 0.6666667 alejandro 3.359166
## 4      kisser    1   capture 2.74084    capture 0.6666667     style 3.359166
## 5      friend    1      sack 2.74084       sack 0.6666667     queen 3.359166
## 6        diss    1    attack 2.74084     attack 0.6666667     rhyme 3.359166
## 7         fun    1   stretch 2.74084    stretch 0.6666667     gonto 3.359166
## 8     capture    1      rack 2.74084       rack 0.6666667     gonna 3.359166
## 9        sack    1    obvious 2.74084   obvious 0.6666667   cheddar 3.359166
## 10      laugh    1     pus-y 2.74084      pus-y 0.6666667      dead 3.359166
```

Co-occurrence visualization

```
source("resources/calculateCoocStatistics.R")
numberOfCoocs <- 10
coocTerm <- "heather"
coocs <- calculateCoocStatistics(coocTerm, binDTM, measure="LOGLIK")
```

```
## Caricamento del pacchetto richiesto: Matrix
```

```
##
## Caricamento pacchetto: 'Matrix'
```

```
## I seguenti oggetti sono mascherati da 'package:tidyr':
##
##     expand, pack, unpack
```

```
print(coocs[1:numberOfCoocs])
```

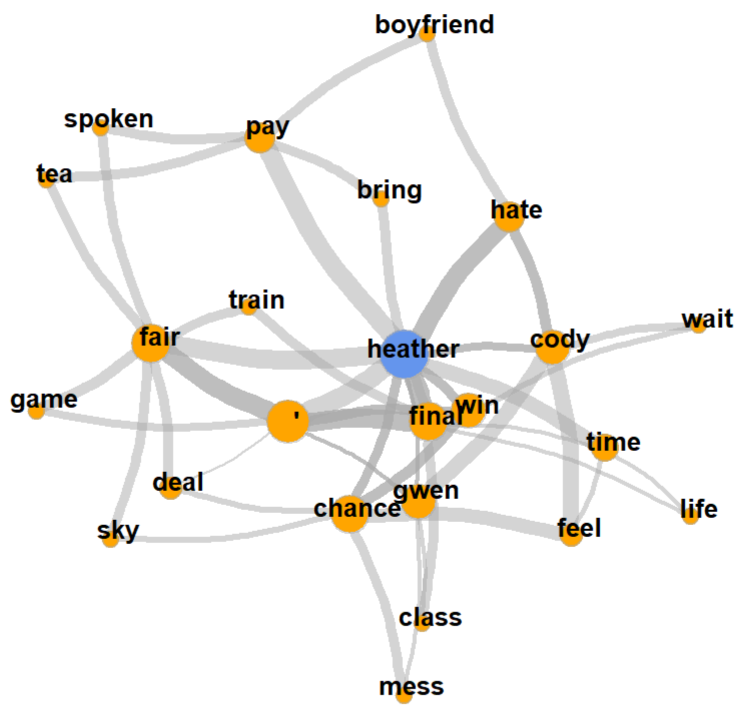
```
##        '     hate    final     fair      pay     time      win   chance
## 3.798286 3.676703 3.676703 3.676703 3.676703 3.038384 2.399123 2.265022
##     cody     gwen
## 2.065521 1.407388
```

```r
resultGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))

# Fill the data.frame to produce the correct number of lines
tmpGraph[1:numberOfCoocs, 3] <- coocs[1:numberOfCoocs]
# Entry of the search word into the first column in all lines
tmpGraph[, 1] <- coocTerm
# Entry of the co-occurrences into the second column of the respective line
tmpGraph[, 2] <- names(coocs)[1:numberOfCoocs]
# Set the significances
tmpGraph[, 3] <- coocs[1:numberOfCoocs]

# Attach the triples to resultGraph
resultGraph <- rbind(resultGraph, tmpGraph)

# Iteration over the most significant numberOfCoocs co-occurrences of the search term
for (i in 1:numberOfCoocs){

  # Calling up the co-occurrence calculation for term i from the search words co-occurrences
  newCoocTerm <- names(coocs)[i]
  coocs2 <- calculateCoocStatistics(newCoocTerm, binDTM, measure="LOGLIK")

  #print the co-occurrences
  coocs2[1:10]

  # Structure of the temporary graph object
  tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
  tmpGraph[1:numberOfCoocs, 3] <- coocs2[1:numberOfCoocs]
  tmpGraph[, 1] <- newCoocTerm
  tmpGraph[, 2] <- names(coocs2)[1:numberOfCoocs]
  tmpGraph[, 3] <- coocs2[1:numberOfCoocs]

  #Append the result to the result graph
  resultGraph <- rbind(resultGraph, tmpGraph[2:length(tmpGraph[, 1]), ])
}

# Sample of some examples from resultGraph
resultGraph[sample(nrow(resultGraph), 6), ]
```

```
##       from      to      sig
## 48  chance heather 2.265022
## 98  chance     sky 1.778608
## 95     pay   bring 2.384113
## 94    fair   train 2.384113
## 86    time   laugh 1.340406
## 109   cody   stick 1.254096
```

```r
# set seed for graph plot
set.seed(1)

# Create the graph object as undirected graph
graphNetwork <- graph.data.frame(resultGraph, directed = F)

# Identification of all nodes with less than 2 edges
verticesToRemove <- V(graphNetwork)[degree(graphNetwork) < 2]
# These edges are removed from the graph
graphNetwork <- delete.vertices(graphNetwork, verticesToRemove)

# Assign colors to nodes (search term blue, others orange)
V(graphNetwork)$color <- ifelse(V(graphNetwork)$name == coocTerm, 'cornflowerblue', 'orange')

# Set edge colors
E(graphNetwork)$color <- adjustcolor("DarkGray", alpha.f = .5)
# scale significance between 1 and 10 for edge width
E(graphNetwork)$width <- scales::rescale(E(graphNetwork)$sig, to = c(1, 10))

# Set edges with radius
E(graphNetwork)$curved <- 0.15
# Size the nodes by their degree of networking (scaled between 5 and 15)
V(graphNetwork)$size <- scales::rescale(log(degree(graphNetwork)), to = c(5, 15))

# Define the frame and spacing for the plot
par(mai=c(0,0,1,0))

# Final Plot
plot(
  graphNetwork,
  layout = layout.fruchterman.reingold, # Force Directed Layout
  main = paste(coocTerm, ' Graph'),
  vertex.label.family = "sans",
  vertex.label.cex = 0.8,
  vertex.shape = "circle",
  vertex.label.dist = 0.5,          # Labels of the nodes moved slightly
  vertex.frame.color = adjustcolor("darkgray", alpha.f = .5),
  vertex.label.color = 'black',     # Color of node names
  vertex.label.font = 2,            # Font of node names
  vertex.label = V(graphNetwork)$name,      # node names
  vertex.label.cex = 1 # font size of node names
)
```

heather Graph



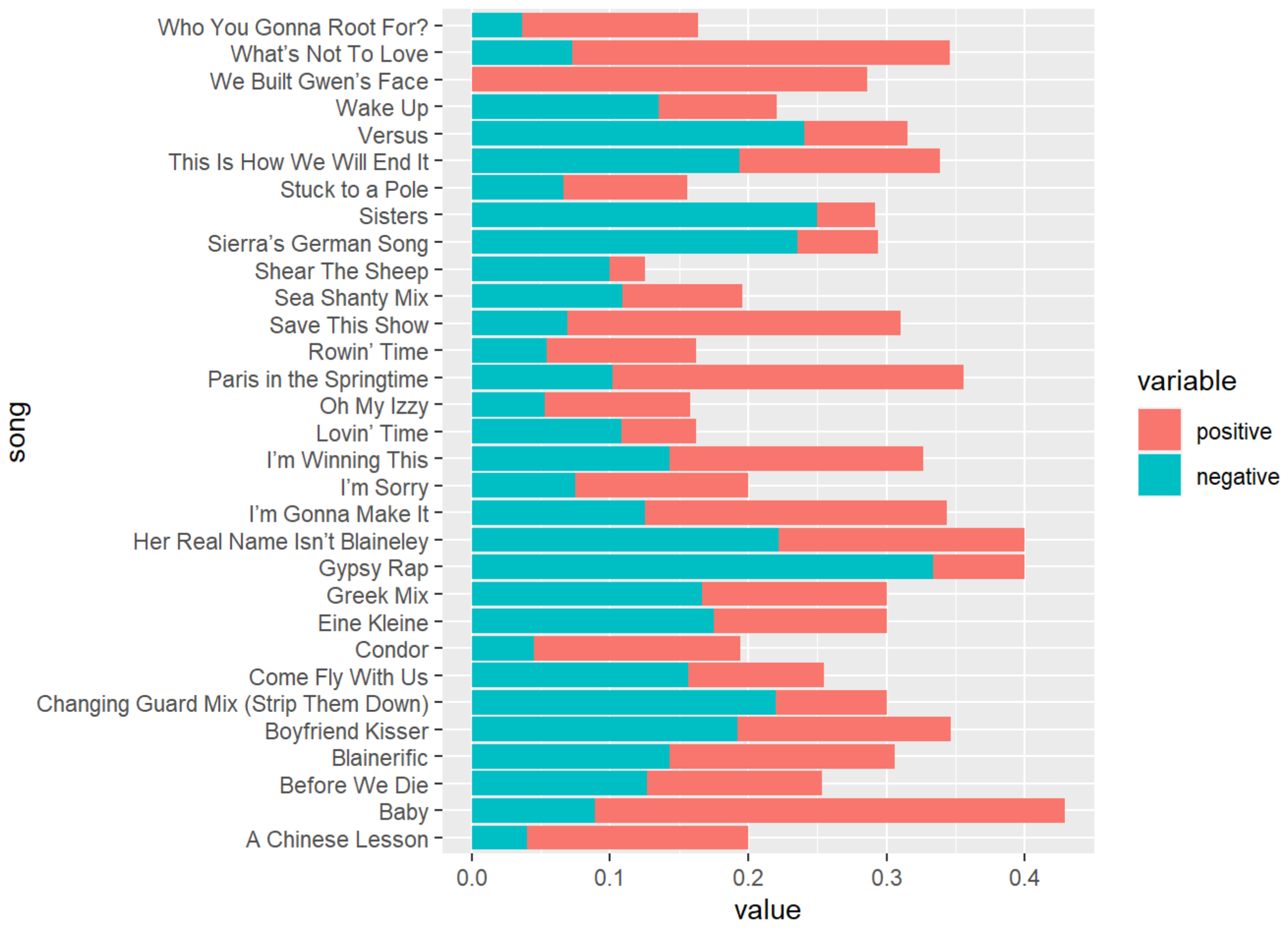
Sentiment analysis

```
sentiment_lexicon <- read.table("resources/NRC-Emotion-Lexicon-Wordlevel-v0.92.txt",
                                header = FALSE, sep = "\t",
                                col.names = c("word", "sentiment", "value"))
sentiment_lexicon_corpus <- sentiment_lexicon %>% filter(word %in% colnames(DTM))
positive_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "positive" & value == 1) %>%
  select(word) %>% pull()
negative_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "negative" & value == 1) %>%
  select(word) %>% pull()
counts_positive <- rowSums(DTM[, positive_terms])
counts_negative <- rowSums(DTM[, negative_terms])
counts_all_terms <- rowSums(DTM)
relative_sentiment_frequencies <- data.frame(
  positive = counts_positive / counts_all_terms,
  negative = counts_negative / counts_all_terms
)
```

```
sentiments_by_song <- aggregate(relative_sentiment_frequencies,
                                 by = list(song = songs$title), mean)

sentiments_by_song %>% head()
```

```
##                                  song  positive   negative
## 1                    A Chinese Lesson 0.1600000 0.04000000
## 2                                Baby 0.3392857 0.08928571
## 3                       Before We Die 0.1267606 0.12676056
## 4                         Blainerific 0.1632653 0.14285714
## 5                    Boyfriend Kisser 0.1538462 0.19230769
## 6 Changing Guard Mix (Strip Them Down) 0.0800000 0.22000000
```

```
df_sentiment <- melt(sentiments_by_song, id.vars = "song")
ggplot(data = df_sentiment, aes(x = song, y = value, fill = variable)) +
  geom_bar(stat="identity", position="stack") + coord_flip()
```



```
positive_songs <- aggregate(
  relative_sentiment_frequencies, by = list(song = songs$title),
  mean) %>% filter(positive > negative)
positive_songs
```

```
##                       song   positive   negative
## 1         A Chinese Lesson 0.16000000 0.04000000
## 2                     Baby 0.33928571 0.08928571
## 3              Blainerific 0.16326531 0.14285714
## 4                   Condor 0.14925373 0.04477612
## 5         I’m Gonna Make It 0.21875000 0.12500000
## 6                 I’m Sorry 0.12500000 0.07500000
## 7         I’m Winning This 0.18367347 0.14285714
## 8               Oh My Izzy 0.10526316 0.05263158
## 9  Paris in the Springtime 0.25423729 0.10169492
## 10             Rowin’ Time 0.10810811 0.05405405
## 11          Save This Show 0.24137931 0.06896552
## 12          Stuck to a Pole 0.08888889 0.06666667
## 13    We Built Gwen’s Face 0.28571429 0.00000000
## 14      What’s Not To Love 0.27272727 0.07272727
## 15 Who You Gonna Root For? 0.12727273 0.03636364
```

```
negative_songs <- aggregate(
  relative_sentiment_frequencies, by = list(song = songs$title),
  mean) %>% filter(negative > positive)
negative_songs
```

```
##                                    song   positive  negative
## 1                       Boyfriend Kisser 0.15384615 0.1923077
## 2  Changing Guard Mix (Strip Them Down) 0.08000000 0.2200000
## 3                       Come Fly With Us 0.09803922 0.1568627
## 4                            Eine Kleine 0.12500000 0.1750000
## 5                              Greek Mix 0.13333333 0.1666667
## 6                              Gypsy Rap 0.06666667 0.3333333
## 7         Her Real Name Isn’t Blaineley 0.17777778 0.2222222
## 8                            Lovin’ Time 0.05405405 0.1081081
## 9                          Sea Shanty Mix 0.08695652 0.1086957
## 10                       Shear The Sheep 0.02500000 0.1000000
## 11                  Sierra’s German Song 0.05882353 0.2352941
## 12                               Sisters 0.04166667 0.2500000
## 13             This Is How We Will End It 0.14516129 0.1935484
## 14                                Versus 0.07407407 0.2407407
## 15                               Wake Up 0.08474576 0.1355932
```

```
neutral_songs <- aggregate(
  relative_sentiment_frequencies, by = list(song = songs$title),
  mean) %>% filter(positive == negative)
neutral_songs
```

```
##            song  positive  negative
## 1 Before We Die 0.1267606 0.1267606
```

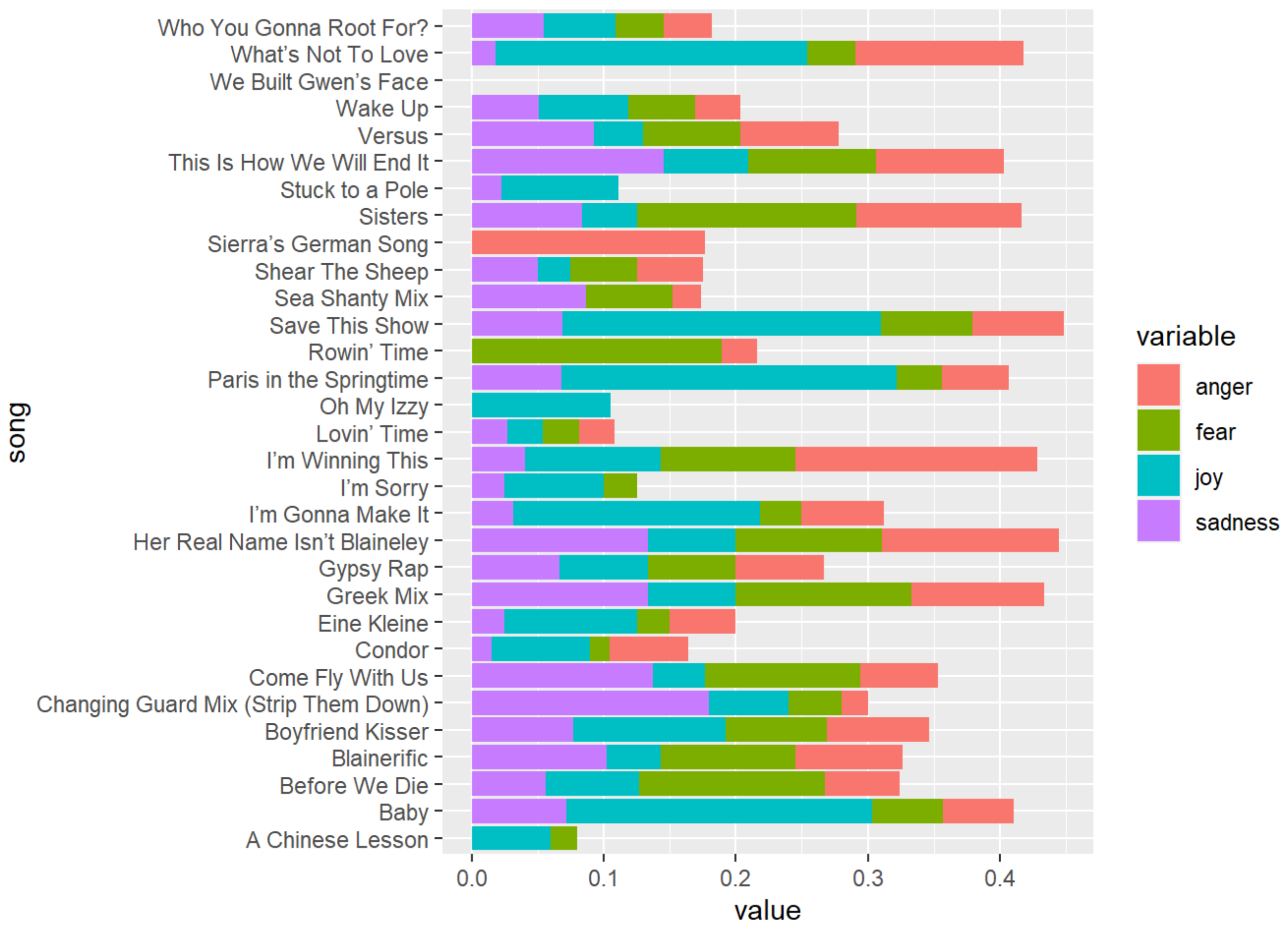
Emotion analysis

```
anger_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "anger" & value == 1) %>%
  select(word) %>% pull()
fear_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "fear" & value == 1) %>%
  select(word) %>% pull()
joy_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "joy" & value == 1) %>%
  select(word) %>% pull()
sadness_terms <- sentiment_lexicon_corpus %>%
  filter(sentiment == "sadness" & value == 1) %>%
  select(word) %>% pull()

counts_anger <- rowSums(DTM[, anger_terms])
counts_fear <- rowSums(DTM[, fear_terms])
counts_joy <- rowSums(DTM[, joy_terms])
counts_sadness <- rowSums(DTM[, sadness_terms])

relative_emotion_frequencies <- data.frame(
  anger = counts_anger / counts_all_terms,
  fear = counts_fear / counts_all_terms,
  joy = counts_joy / counts_all_terms,
  sadness = counts_sadness / counts_all_terms
)
```

```
emotions_by_song <- aggregate(relative_emotion_frequencies,
                              by = list(song = songs$title), mean)

head(emotions_by_song)
```

```
##                                  song      anger       fear        joy
## 1                    A Chinese Lesson 0.00000000 0.02000000 0.06000000
## 2                                Baby 0.05357143 0.05357143 0.23214286
## 3                      Before We Die 0.05633803 0.14084507 0.07042254
## 4                         Blainerific 0.08163265 0.10204082 0.04081633
## 5                    Boyfriend Kisser 0.07692308 0.07692308 0.11538462
## 6 Changing Guard Mix (Strip Them Down) 0.02000000 0.04000000 0.06000000
##      sadness
## 1 0.00000000
## 2 0.07142857
## 3 0.05633803
## 4 0.10204082
## 5 0.07692308
## 6 0.18000000
```

```
df_emotions <- melt(emotions_by_song, id.vars = "song")
ggplot(data = df_emotions, aes(x = song, y = value, fill = variable)) +
  geom_bar(stat="identity", position="stack") + coord_flip()
```


Who You Gonna Root For?
What's Not To Love
We Built Gwen's Face
Wake Up
Versus
This Is How We Will End It
Stuck to a Pole
Sisters
Sierra's German Song
Shear The Sheep
Sea Shanty Mix
Save This Show
Rowin' Time
Paris in the Springtime
Oh My Izzy
Lovin' Time
I'm Winning This
I'm Sorry
I'm Gonna Make It
Her Real Name Isn't Blaineley
Gypsy Rap
Greek Mix
Eine Kleine
Condor
Come Fly With Us
Changing Guard Mix (Strip Them Down)
Boyfriend Kisser
Blainerific
Before We Die
Baby
A Chinese Lesson
song
0.0
0.1
0.2
0.3
0.4
value
variable
anger
fear
joy
sadness