Using STED

2022-10-22

Loading data

Data sources have been chosen such that they are expected to have low and high potential for meaningful entertainment experiences respectively. The first is Hansard (record of UK Parliament) and the second is Eurovision Song Contest lyrics.

https://hansard.parliament.uk/commons/2022-10-20 https://www.kaggle.com/datasets/minitree/eurovision-song-lyrics?resource=download

Hansard

Data has been retrieved for the debates in the lower house in October 2022. With more care, the words of each individual could be identified separately. However, here each day is considered separately.

```
filenames <- tibble('long' = list.files(path = './hansard_data/', pattern="*.txt", full.names = T),
                    'short' = list.files(path = './hansard_data', pattern="*.txt"))

hansard_raw <- tibble(date = as.Date(character()),
                  text = character())

for (i in 1:length(filenames$long)){
  new_text = read_file(filenames$long[i])
  new_row = tibble(
    date = filenames$short[i] %>% str_remove("Commons Chamber ") %>% str_remove(".txt") %>% as.Date(),
    text = new_text)
  hansard_raw <- bind_rows(hansard_raw, new_row)}

hansard <-
  corpus(hansard_raw,
         docid_field = "date",
         text_field = "text") %>%
  quanteda::tokens(remove_punct = TRUE,
                   remove_numbers = TRUE,
                   remove_symbols = TRUE,
                   remove_url = TRUE) %>%
  tokens_tolower() %>%
  tokens_remove(stopwords("english")) %>%
  tokens_wordstem()
```

Eurovision lyrics

The lyrics are provided in their original language and with a translation too. Here the English translation will be used where the original song was in another language.

```
esc_raw <- jsonlite::fromJSON(txt="eurovision-lyrics-2022.json")

esc_data <- tibble(song_id = character(),
```

```
                    country = character(),
                    name = character(),
                    language = character(),
                    year = character(),
                    lyrics = character())

for (i in 1:length(esc_raw)){
  new_data = esc_raw[[i]]
  new_row = tibble(song_id = new_data[[1]],
                    country = new_data[[2]],
                    name = new_data[[5]],
                    language = new_data[[6]],
                    year = new_data[[10]],
                    lyrics = if_else(new_data[[14]]=="English", new_data[[13]], new_data[[14]]))
  esc_data <- bind_rows(esc_data, new_row)}

esc <-
  corpus(esc_data,
         docid_field = "song_id",
         text_field = "lyrics") %>%
  quanteda::tokens(remove_punct = TRUE,
                    remove_numbers = TRUE,
                    remove_symbols = TRUE,
                    remove_url = TRUE) %>%
  tokens_tolower() %>%
  tokens_remove(stopwords("english")) %>%
  tokens_wordstem()
```

STED

Hansard

```
sted_dict <- dictionary(file = "pone.0239050.s003.dic", format = "LIWC")

hansard_sted <-
  tokens_lookup(hansard, dictionary = sted_dict) %>%
  dfm() %>%
  dfm_group()

hansard_sted_df <- convert(hansard_sted, to = "data.frame")

hansard_all_df <- hansard %>%
  ntoken() %>%
  as.data.frame() %>%
  rownames_to_column(var = "doc_id") %>%
  rename("tokens" = ".")

hansard_final <-
  hansard_sted_df %>%
  left_join(hansard_all_df, by = 'doc_id') %>%
  mutate(general = general/tokens,
         awe = awe/tokens,
         gratitude = gratitude/tokens,
```

```
         elevation = elevation/tokens,
         admiration = admiration/tokens,
         hope = hope/tokens,
         doc_id = as.Date(doc_id)) %>%
  select(-tokens) %>%
  pivot_longer(
    cols = !doc_id)
```

Eurovision STED

```
esc_sted <-
  tokens_lookup(esc, dictionary = sted_dict) %>%
  dfm() %>%
  dfm_group(group = year)

esc_sted_df <- convert(esc_sted, to = "data.frame")
esc_all_df <-
  esc %>%
  tokens_group(groups = year) %>%
  ntoken() %>%
  as.data.frame() %>%
  rownames_to_column(var = "doc_id") %>%
  rename("tokens" = ".")

esc_final <-
  esc_sted_df %>%
  left_join(esc_all_df, by = 'doc_id') %>%
  mutate(general = general/tokens,
         awe = awe/tokens,
         gratitude = gratitude/tokens,
         elevation = elevation/tokens,
         admiration = admiration/tokens,
         hope = hope/tokens,
         doc_id = as.numeric(doc_id)) %>%
  select(-tokens) %>%
  pivot_longer(
    cols = !doc_id)
```

Plots

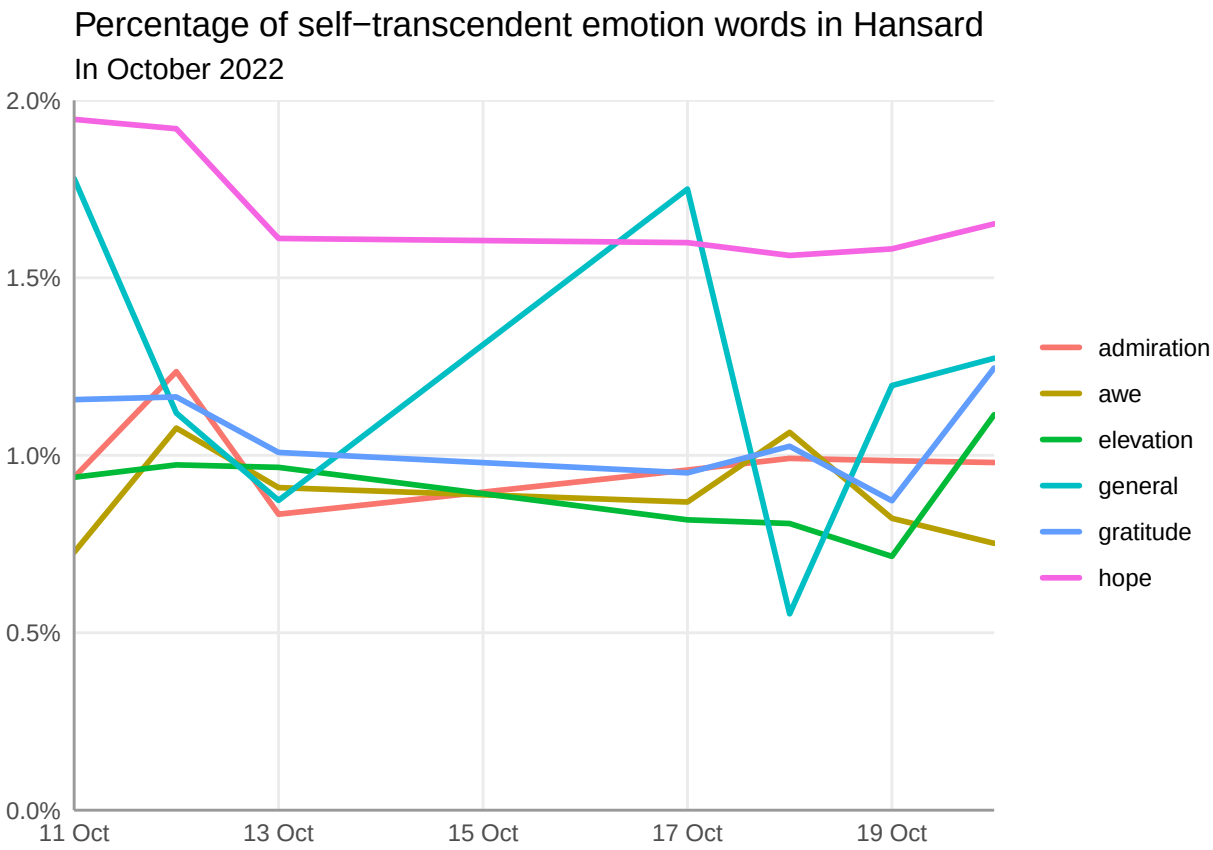
Hansard

```
ggplot(hansard_final, aes(x = doc_id, y = value, group = name, colour = name)) +
  geom_line(size=1, lineend="round") +
  scale_y_continuous(
    limits = c(0,0.02),
    labels = label_percent(),
    expand = c(0,0)) +
  scale_x_date(
    date_labels = "%d %b",
    expand = c(0,0)) +
  theme_minimal() +
  theme(legend.title=element_blank(),
```

```
        axis.title = element_blank(),
        axis.line = element_line(size = 0.5, colour = "grey60"),
        panel.grid.minor = element_blank()) +
  labs(
    title = "Percentage of self-transcendent emotion words in Hansard",
    subtitle = "In October 2022")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

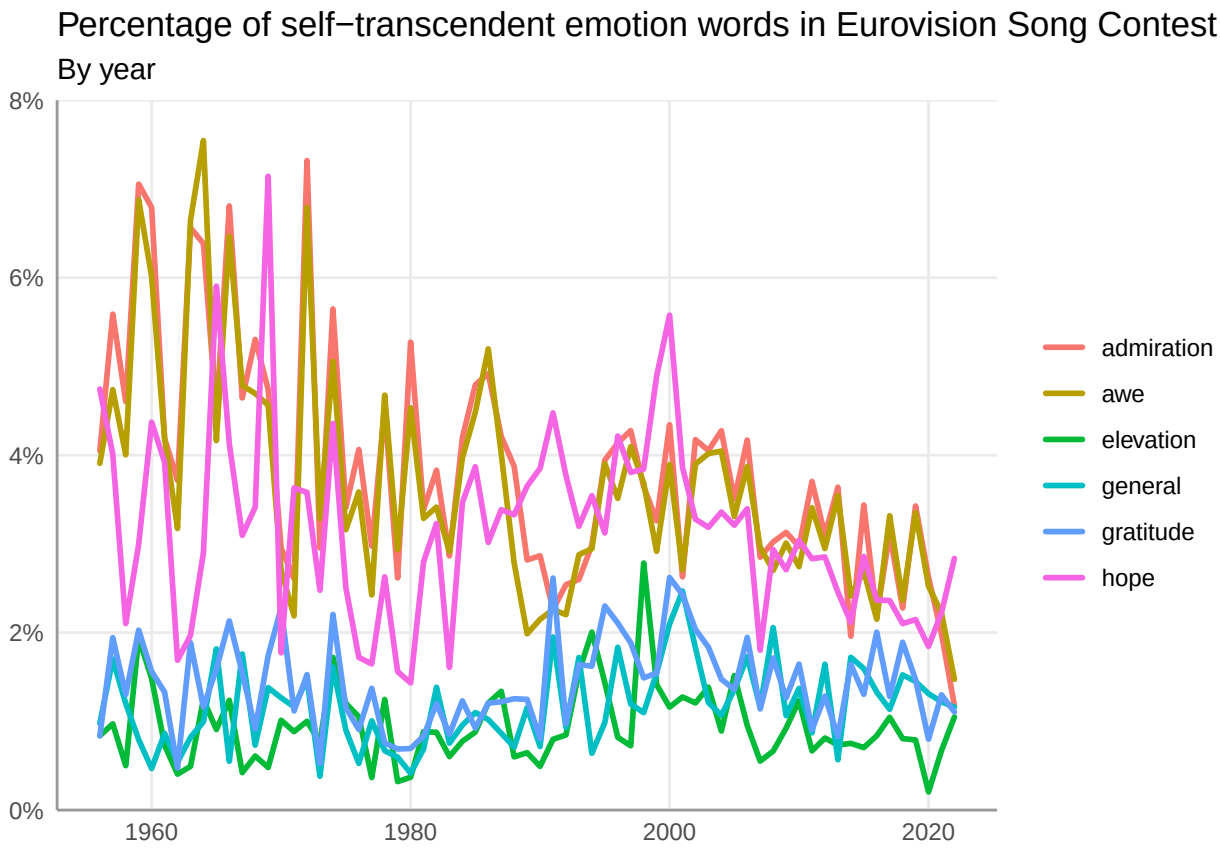
```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Eurovision

```
ggplot(esc_final, aes(x = doc_id, y = value, group = name, colour = name)) +
  geom_line(size=1, lineend="round") +
  scale_y_continuous(
    limits = c(0,0.08),
    labels = label_percent(),
    expand = c(0,0)) +
```

```
  theme_minimal() +
  theme(legend.title=element_blank(),
        axis.title = element_blank(),
        axis.line = element_line(size = 0.5, colour = "grey60"),
        panel.grid.minor = element_blank()) +
  labs(
    title = "Percentage of self-transcendent emotion words in Eurovision Song Contest lyrics",
    subtitle = "By year")
```



Examples

Individual countries' Eurovision songs

```
esc$year_country <- paste0(esc$year, " ", esc$country)

esc_sted2 <-
  tokens_lookup(esc, dictionary = sted_dict) %>%
  dfm() %>%
  dfm_group(group = year_country)

esc_sted_df2 <-
  convert(esc_sted2, to = "data.frame") %>%
  mutate(year = as.numeric(str_extract(doc_id, "[0-9]{4}")),
         country = str_remove(doc_id, "[0-9]{4} ")) %>%
  select(-doc_id)
esc_all_df2 <-
```

```
  esc %>%
  tokens_group(groups = year_country) %>%
  ntoken() %>%
  as.data.frame() %>%
  rownames_to_column(var = "doc_id") %>%
  rename("tokens" = ".") %>%
  mutate(year = as.numeric(str_extract(doc_id, "[0-9]{4}")),
         country = str_remove(doc_id, "[0-9]{4} ")) %>%
  select(-doc_id)

esc_final2 <-
  esc_sted_df2 %>%
  left_join(esc_all_df2, by = c('year', 'country')) %>%
  mutate(general = general/tokens,
         awe = awe/tokens,
         gratitude = gratitude/tokens,
         elevation = elevation/tokens,
         admiration = admiration/tokens,
         hope = hope/tokens) %>%
  select(-tokens) %>%
  pivot_longer(
    cols = !c(year, country)) %>%
  mutate(country = case_when(country=="The Netherlands" ~ "Netherlands",
                             str_detect(country, "Germany") ~ "Germany",
                             T ~ country))
```

```
esc_final2 %>%
  filter(country %in% c("Germany", "United Kingdom")) %>%
ggplot(aes(x = year, y = value, group = country, colour = country)) +
  geom_line() +
  scale_y_continuous(
    limits = c(0,0.2),
    labels = label_percent(),
    expand = c(0,0)) +
  theme_minimal() +
  theme(legend.title=element_blank(),
        axis.title = element_blank(),
        axis.line = element_line(size = 0.5, colour = "grey60"),
        panel.grid.minor = element_blank()) +
  labs(title = "Percentage of self-transcendent emotion words in Eurovision Song Contest lyrics",) +
  facet_wrap(~ name, nrow = 3)
```

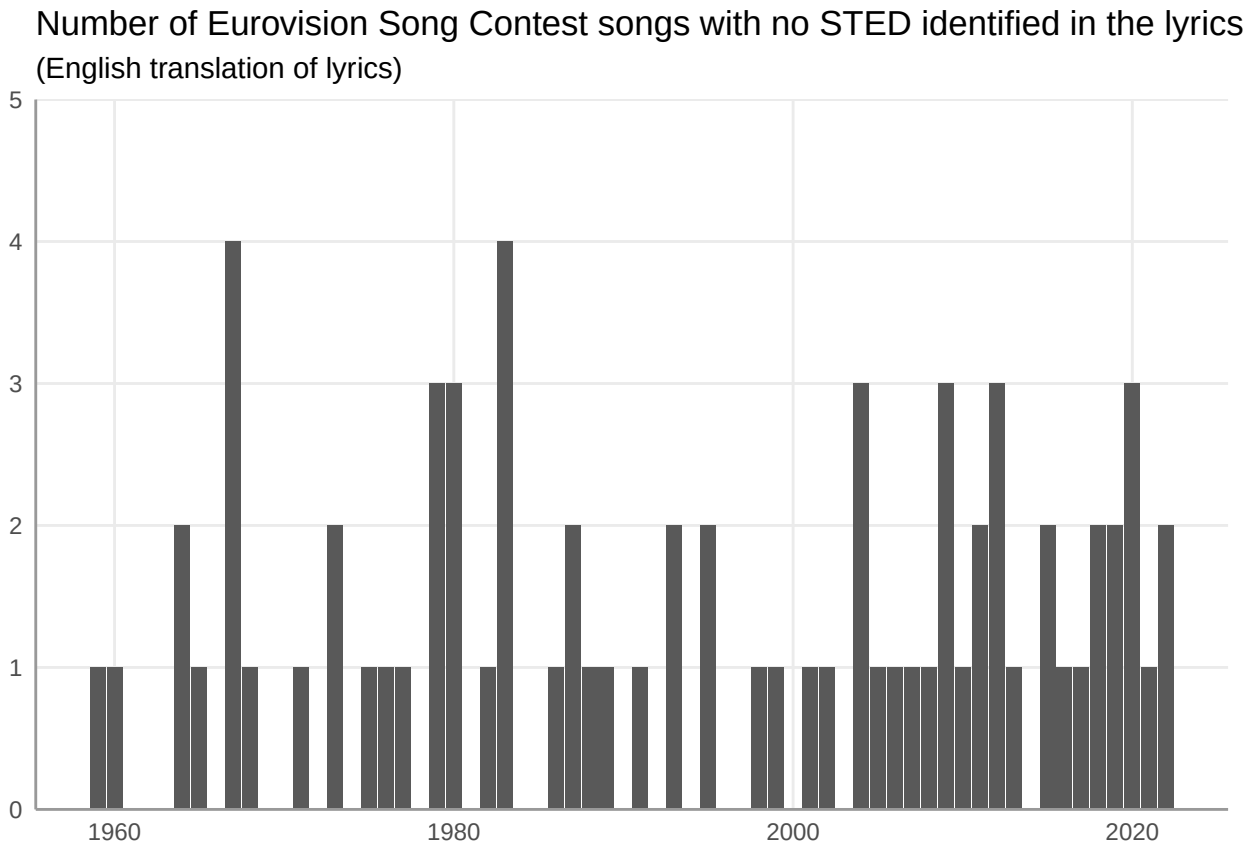
Germany vs UK

Songs with no STED identified

```
esc_no_sted <-
  esc_final2 %>%
  group_by(year, country) %>%
  summarise(sum = sum(value)) %>%
  filter(sum==0)
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
ggplot(esc_no_sted, aes(x = year)) +
  geom_bar() +
  theme_minimal() +
  scale_y_continuous(
    expand = c(0,0),
    limits = c(0,5)) +
  theme(legend.title=element_blank(),
        axis.title = element_blank(),
        axis.line = element_line(size = 0.5, colour = "grey60"),
        panel.grid.minor = element_blank()) +
  labs(
    title = "Number of Eurovision Song Contest songs with no STED identified in the lyrics",
    subtitle = "(English translation of lyrics)")
```



Summary

As expected, the rate of self-transcendent emotion words was much lower in the Hansard records of the UK Parliament than in Eurovision song lyrics. In parliament, hope was generally the most identified self-transcendent emotion, but all of the emotion types remained at below 2% of all words (after stop word removal). In contrast, many Eurovision songs had values above 15%. There is significant variation across songs but most have more awe, admiration and hope than elevation, general or gratitude.

72 of the songs have not been labelled with any self-transcendent emotions. These include "Der K und K Kalypso aus Wien" for Austria in 1959 with lyrics including "The Calypso from Vienna. It's not from Ecuador. It's not from Spain." and "Higher Ground" for Denmark in 2018 with lyrics including "Freeze the arrow in the air. Make your mark and leave it hanging there."