DATA 607 TidyVerse Vignette Create Assignment

Text Analysis and Animation with tidytext and gganimate

Kevin Kirby

2024-10-18

Overview

In this vignette, I'll show how to combine `tidytext` for text mining with `gganimate` for animated visuals to create dynamic representations of text data. I'll be using a data set I have includes track metadata and artist biographies from Beatport.com . After ingesting and pre-processing using `tidytext`, I'll run a sentiment analysis, using `gganimate` to bring the results to life in GIF and MP4 video.

The question I want to answer during this exercise is: has the sentiment of artist biographies associated with different music genres changed over time?

Install and Load Required Packages

These are the required libraries. Please take care to install the ones you do not have.

```
#parquet file reader
library(arrow)
```

```
##
## Attaching package: 'arrow'
```

```
## The following object is masked from 'package:utils':
##
##     timestamp
```

```
#tidyverse data wrangling
library(tidytext)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::duration() masks arrow::duration()
## x dplyr::filter()       masks stats::filter()
## x dplyr::lag()          masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(textdata)

#visualization and animation
library(ggplot2)
```

```
library(gganimate)
library(dplyr)

#these three will let you render and view MP4 and gifs
library(av)
library(gifski)
library(magick)
```

```
## Linking to ImageMagick 6.9.12.93
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

Loading the data

I uploaded a `parquet` cache to my GCP instance and then set the link to public download. A parquet is a smaller file that's a cached representation of the full data file. This allowed me to reduce a 2GB file down to .5 GB, making it easier to share.

```
gcp_bp_tidy_url <- "https://storage.googleapis.com/data_science_masters_files/2024_fall/data_607_data_ma

bp_tidy_temp <- tempfile(fileext = ".parquet")

download.file(gcp_bp_tidy_url, bp_tidy_temp, mode = "wb")

bp_tidy_df <- read_parquet(bp_tidy_temp)
```

Tidying the data

I want to check that the date column is actually a date and then drop anything before 2018. This will let me focus on data from more recent years.

```
bp_tidy_df <- bp_tidy_df %>%
  mutate(release_date = as.Date(release_date)) %>%
  filter(!is.na(beatport_bio) & beatport_bio != "")

bp_tidy_df <- bp_tidy_df[bp_tidy_df$release_date >= as.Date("2018-01-01"), ]
```

Sentiment analysis

To perform a sentiment analysis, the text values need to be turned into tokens. The tokenization function `unnest_tokens` comes from the `tidytext` universe and lets you break out each token into its own row. I've used the default "words" setting, where each word becomes a token. Other options include things like characters, sentences, or lines.

The actual assessment of what this means will be done with the aid of the National Research Council of Canada's (NRC) Word-Emotion Association Lexicon. It assigns 0 or 1 values for the below sentiments and emotions based on whether the world does or does not associate with it.

sentiments: * negative * positive

emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, trust

To tie the tokens to the NRC, you can do a "many-to-many" join on the word field. Each word can appear multiple times and be linked to different NRC categories. The `count()` function groups the data release year, genre, and sentiment.

```
bp_bio_tokens <- bp_tidy_df %>%
  unnest_tokens(word, input = beatport_bio)

nrc_lex <- get_sentiments("nrc")

bp_bio_nrc <- bp_bio_tokens %>%
  inner_join(nrc_lex, by = "word", relationship = "many-to-many") %>%
  count(release_year = year(release_date), genre_name, sentiment) %>%
  spread(sentiment, n, fill = 0)
```

Next, I normalized the data by calculating two key metrics: the Emotional Polarity Index (EPI) and Sentiment Score.

EPI: This provides a more holistic view by summing all positive emotions and subtracting negative ones, which gives a broader perspective on emotional tone beyond a simple positive/negative split.

Sentiment Score: A more traditional metric, calculated by subtracting the negative emotions from the positive ones, offering a straightforward comparison of positive versus negative sentiment.

I normalized each sentiment category by dividing its value by the total of all emotions and then multiplying by 100. This converts the counts into percentages, making it easier to see how each sentiment contributes relative to the overall emotional expression across different genres and years.

```
bp_bio_nrc <- bp_bio_nrc %>%
  mutate(all_emotions = anticipation + joy + surprise + trust + disgust + fear + sadness) %>%
  mutate(anticipation = anticipation / all_emotions * 100,
         joy = joy / all_emotions * 100,
         surprise = surprise / all_emotions * 100,
         trust = trust / all_emotions * 100,
         disgust = disgust / all_emotions * 100,
         fear = fear / all_emotions * 100,
         sadness = sadness / all_emotions * 100,
         positive = positive / all_emotions * 100,
         negative = negative / all_emotions * 100)

bp_bio_nrc <- bp_bio_nrc %>%
  mutate(EPI = (anticipation + joy + surprise + trust) - (disgust + fear + sadness)) %>%
  mutate(sentiment_score = positive - negative)
```

Using gganimate to bring the data to life

With normalized sentiment metrics in hand, we can bring them to life with animated visualizations. `gganimate` allows us to render traditional `ggplot2` charts in more compelling ways, like a GIF or an MP4 video. The goal is to have the eyes drawm towards anything that stands out.

To demonstrate, I selected a subset of genres that my personal favorites:

```
bp_bio_nrc_lf <- bp_bio_nrc %>%
  select(release_year, genre_name, EPI, sentiment_score) %>%
  gather(key = "measure", value = "value", EPI, sentiment_score)%>%
  filter(genre_name %in% c("Melodic House & Techno",
                           "Afro House",
                           "Organic House / Downtempo",
                           "Progressive House",
                           "Techno (Raw / Deep / Hypnotic)"))
```

To demonstrate, I'm using `facet_wrap` to create side-by-side charts that show the EPI and Sentiment Score

trends by genre and year. The `ggplot` chart is saved as a variable and animated with `transition_reveal()`, which lets the lines and points gradually appear as the years progress. This makes it easier to spot how each genre's sentiment and EPI evolve over time. The final step generates both a GIF and an MP4 video, using `gifski_renderer` for the GIF and `av_renderer` for the video.

```
bp_bio_nrc_viz <- ggplot(bp_bio_nrc_lf, aes(x = release_year, y = value, color = genre_name, group = ge
  geom_line(size = 1.2) +
  geom_point(aes(size = abs(value))) +
  scale_color_viridis_d(option = "C")  +
  labs(title = "EPI & Sentiment Score By Genre and Year",
       y = "Value",
       x = "Year") +
  facet_grid(~ measure, scales = "free_y") +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.title = element_text(size = 14),
        legend.text = element_text(size = 12)) +
  guides(size = "none") +
  transition_reveal(release_year) +
  ease_aes('linear')
```

```
## 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.
```

```
if (interactive()) {
  animate(
    bp_bio_nrc_viz,
    width = 1024,
    height = 768,
    nframes = 600,
    fps = 60,
    duration = 10,
    renderer = gifski_renderer(file = "bp_bio_nrc_viz.gif")
  )

  animate(
    bp_bio_nrc_viz,
    width = 1024,
    height = 768,
    nframes = 600,
    fps = 60,
    duration = 10,
    renderer = av_renderer(file = "bp_bio_nrc_viz.mp4")
  )
}
```

Conclusion

Data visualizations should be used to aid data understanding. Data is truth and truth must win out.