Lyrical Analysis Exploration

2025-03-15

Load Libraries

```
library(class)
library(tm)
library(MASS)
library(nnet)
library(tidyverse)
library(caret)
library(tidytext)
library(olsrr)
```

Load Data

```
# Read Lines
lines <- readLines("lyran.csv", encoding = "UTF-8", warn = FALSE)
```

lyran.csv is the csv file of the song titles, artists, and lyrics

```
## Incomplete Final Line Warning
# Get only valid lines
cleaned_lines <- iconv(lines, from = "UTF-8", to = "UTF-8", sub = "byte")
valid_lines <- cleaned_lines[!is.na(cleaned_lines)]
# Write Cleaned to New File
writeLines(valid_lines, "lyran_cleaned.csv")

# Read the Cleaned File
lyrics_data <- read.csv("lyran_cleaned.csv", fileEncoding = "UTF-8", stringsAsFactors = FALSE)
```

Stop Words and Cleaning Data

```
# Define your custom stop words
custom_stopwords <- c("ah", "im", "ohohoh", "oo", "uhhuh", "nooh", "s", "yearold", "thatll", "hadnt", "
# Combine custom stopwords with the default (english) stop words
all_stopwords <- c(stopwords("en"), custom_stopwords)
corpus <- Corpus(VectorSource(lyrics_data$lyrics))
# Clean the corpus
corpus_clean <- corpus %>%
  tm_map(tolower) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, all_stopwords) %>%
  tm_map(stripWhitespace)
```

```r
tidy_lyrics <- data.frame(
  name = rep(lyrics_data$name, each = sapply(corpus_clean, length)),
  artist = rep(lyrics_data$artist, each = sapply(corpus_clean, length)),
  theme = rep(lyrics_data$theme, each = sapply(corpus_clean, length)),
  text = sapply(corpus_clean, as.character),
  row_id = seq_along(lyrics_data$name),  # Track the original row order
  stringsAsFactors = FALSE
)
```

Reformatting Data

```r
# Make Unique Song_Artist_Id
lyrics_data <- tidy_lyrics %>%
  mutate(song_artist_id = paste(name, artist, sep = " ")) %>%
  dplyr::select(song_artist_id, theme, text)
```

Count Words

```r
lyrics_data_tokens <- lyrics_data %>%
  mutate(text = as.character(text)) %>% # Ensure text is character
  unnest_tokens(output = word, input = text) %>% # Tokenize
  group_by(song_artist_id, theme, word) %>%
  summarise(word_count = n(), .groups = "drop") # Count words
```

Pivot

```r
lyrics_data_wide <- lyrics_data_tokens %>%
  pivot_wider(names_from = word,
              values_from = word_count,  # Use word_indicator instead of word_count
              values_fill = 0,  # Fill missing values with 0
              names_glue = "word_{word}",  # Custom column naming for words
              names_repair = "unique")  # Make column names unique
```

Log Transform Then Do Within Column

```r
lyrics_data_log_norm <- lyrics_data_wide %>%
  mutate(across(starts_with("word_"), ~ log(. + 1))) %>%  # Log-transform
  mutate(across(starts_with("word_"), ~ (. - min(.)) / (max(.) - min(.))))  # Normalize
```

Pivot Theme

```r
lyrics_theme_binary <- lyrics_data_tokens %>%
  group_by(song_artist_id, theme) %>%
  summarise(theme_indicator = 1, .groups = "drop")
lyrics_theme_onehot <- lyrics_theme_binary %>%
  pivot_wider(
    names_from = theme,
```

```
    values_from = theme_indicator,
    values_fill = 0,
    names_glue = "{theme}_theme" # Custom column naming for themes
  )
```

```
lyrics_data_log_norm <- as.data.frame(lyrics_data_log_norm)

lyrics_data_log_norm_factor <- lyrics_data_log_norm %>% mutate(theme_as_factor = factor(theme, ordered =
```

```
lyrics_combined <- full_join(lyrics_theme_onehot, lyrics_data_log_norm_factor, by = join_by(song_artist,
```

Filter to Help Pick Words

```
lyrics_data_log_norm %>%
  filter(theme == "crush") %>%  # Filter for the 'crush' theme
  pivot_longer(cols = -c(song_artist_id, theme), names_to = "word", values_to = "value") %>%  # Pivot t
  filter(value > 0) %>%  # Only include rows where the word appears (value > 0)
  group_by(song_artist_id, word) %>%
  summarise(word_value_in_song = sum(value), .groups = "drop") %>%  # Count how many times the word app
  group_by(word) %>%
  summarise(
    total_word_value = sum(word_value_in_song), # Total word occurrences across all songs
    sd = sd(word_value_in_song),
    unique_songs = n_distinct(song_artist_id),  # Number of unique songs the word appears in
    .groups = "drop"
  ) %>%
  filter(unique_songs>1) %>%
  arrange(desc(total_word_value))  # Sort by total word count in descending order
```

```
## # A tibble: 381 x 4
##    word       total_word_value    sd unique_songs
##    <chr>                 <dbl> <dbl>        <int>
##  1 word_just             10.4  0.161           24
##  2 word_know              9.71 0.189           24
##  3 word_like              8.97 0.169           23
##  4 word_yeah              8.35 0.172           21
##  5 word_make              7.43 0.208           13
##  6 word_wanna             6.73 0.188           16
##  7 word_want              6.73 0.186           17
##  8 word_baby              6.46 0.118           18
##  9 word_now               6.39 0.258           14
## 10 word_tell              5.50 0.146           12
## # i 371 more rows
```

Testing Words

```
word_lm <- lm(word_yeah ~ rebellion_theme + love_theme + `moving on_theme` + growth_theme + situationshi
anova_result <- anova(word_lm)
anova_summary <- broom::tidy(anova_result)
# Filter only significant variables (p-value < 0.05)
significant_results <- anova_summary %>%
  arrange(p.value) %>%
```

```
  filter(p.value < 0.0001)
# Display results
print(significant_results)
```

```
## # A tibble: 1 x 6
##   term           df sumsq meansq statistic   p.value
##   <chr>       <int> <dbl>  <dbl>     <dbl>     <dbl>
## 1 happy_theme     1 0.704  0.704      17.0 0.0000422
```

```
df_metrics <- read_csv("~/Desktop/SDS 293 Data (Copy of MTH 354) /Machine Learning/MTH 354 Work.csv")
```

```
write.csv(lyrics_combined, "lyrics_combined.csv")
```