Lyrical Analysis

2025-03-15

Load Libraries

```r
library(class)
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.3.3
```

```
## Loading required package: NLP
```

```
## Warning: package 'NLP' was built under R version 4.3.3
```

```r
library(MASS)
library(nnet)
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   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x dplyr::select()     masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(tidyr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tidytext)
```

Load Data

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

```
## Warning in tm_map.SimpleCorpus(., tolower): transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(., removePunctuation): transformation drops
## documents
```

```
## Warning in tm_map.SimpleCorpus(., removeNumbers): transformation drops
## documents
```

```
## Warning in tm_map.SimpleCorpus(., removeWords, all_stopwords): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(., stripWhitespace): transformation drops
## documents
```

```
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
)
```

```
## Warning in rep(lyrics_data$name, each = sapply(corpus_clean, length)): first
## element used of 'each' argument
```

```
## Warning in rep(lyrics_data$artist, each = sapply(corpus_clean, length)): first
## element used of 'each' argument

## Warning in rep(lyrics_data$theme, each = sapply(corpus_clean, length)): first
## element used of 'each' argument
```

Reformatting Data

```
# 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

```
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

```
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
```

Normalize Within Column

```
# Normalize each word column within the column (Min-Max Scaling)
lyrics_data_normalized <- lyrics_data_wide %>%
  mutate(across(starts_with("word_"), ~ (. - min(.)) / (max(.) - min(.)), .names = "norm_{.col}")) %>%
  dplyr::select(-starts_with("word_"))
```

Log Transform Then Do Within Column

```
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

```
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 == "age/power dynamic") %>%  # Filter for the 'age/power dynamic' 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: 358 x 4
##    word       total_word_value     sd unique_songs
##    <chr>                 <dbl>  <dbl>        <int>
##  1 word_never             8.29 0.188            19
##  2 word_know              7.99 0.196            18
##  3 word_now               6.70 0.165            17
##  4 word_just              6.66 0.137            16
##  5 word_like              5.65 0.153            18
##  6 word_love              5.52 0.184            13
##  7 word_baby              4.27 0.209            10
##  8 word_tell              4.06 0.167             9
##  9 word_cause             3.79 0.151            11
## 10 word_time              3.71 0.0898            8
## # i 348 more rows
```

Testing and Training Data

```
lyrics_data_log_norm_factor <- as.data.frame(lyrics_data_log_norm_factor)
lc_size <- floor(0.75 * nrow(lyrics_combined))
```

```
# Set Seed
set.seed(200)
train_ind_lc <- sample(seq_len(nrow(lyrics_combined)), size = lc_size)

train_lc <- lyrics_combined[train_ind_lc, ]
test_lc <- lyrics_combined[-train_ind_lc, ]

train_ldn <- lyrics_data_log_norm_factor[train_ind_lc, ]
test_ldn <- lyrics_data_log_norm_factor[-train_ind_lc, ]
```

LDA Fit

```
lda.fit <- lda(theme_as_factor ~ word_afford, data = lyrics_data_log_norm_factor, subset = train_ind_lc
#print(lda.fit)
```

LDA Results

```
lda.pred <- predict(lda.fit, newdata = test_ldn)
lda.class <- lda.pred$class
lda_matrix <- table(Predicted = lda.class, Actual = test_ldn$theme_as_factor)
#print(lda_matrix)
actual_classes <- test_ldn$theme_as_factor
accuracy <- mean(lda.class == actual_classes)
print(paste("Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy: 3.18 %"
```

```
conf_matrix <- table(Predicted = lda.class, Actual = actual_classes)
#print(conf_matrix)
theme_accuracies <- diag(conf_matrix) / rowSums(conf_matrix)
theme_accuracies <- round(theme_accuracies * 100, 2)  # Convert to percentage
print(theme_accuracies)
```

```
##  age/power dynamic              crush        empowerment               exes
##               0.00                NaN                NaN                NaN
##     forbidden love              grief         growing up             growth
##                NaN                NaN                NaN                NaN
##              happy               hate         heartbreak           jealousy
##                NaN                NaN                NaN                NaN
##               love      mental health          moving on           partying
##                NaN                NaN               3.23                NaN
##          rebellion           religion        reminiscing            revenge
##                NaN                NaN                NaN                NaN
##      situationship toxic relationship         unrequited
##                NaN                NaN                NaN
```

Testing Words

```
word_lm <- lm(word_yeah ~ `age/power dynamic_theme` + rebellion_theme + love_theme + `moving on_theme`
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: 2 x 6
##   term               df sumsq meansq statistic   p.value
##   <chr>           <int> <dbl>  <dbl>     <dbl>     <dbl>
## 1 happy_theme         1 0.704  0.704      17.2 0.0000381
## 2 partying_theme      1 0.628  0.628      15.4 0.0000986
```

Feature Selection

Correlation Based Feature Selection

```
#X <- train_lc[,-c(1:24,7332)]
#Y <- train_lc$theme_as_factor
```

Recursive Feature Elmination

```
#control <- rfeControl(functions = rfFuncs, method = "cv", number = 5)
#rfe_result <- rfe(X, Y, sizes = c(1:10), rfeControl = control)

# Selected features
#X_selected <- X[, predictors(rfe_result)]
```

```
# Remove them
#X_filtered <- X[, -highly_correlated]
```

Multi Log

```
multi_log_model <- multinom(theme_as_factor ~ word_love + word_baby +
    word_yeah + word_still + word_never,
                            data = train_lc, trace = FALSE)
multi_log_pred <- predict(multi_log_model)#, newdata = test_lc)
# Calculate Accuracy
accuracy <- mean(multi_log_pred == train_lc$theme_as_factor)
print(paste("Multinomial Logistic Regression Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Multinomial Logistic Regression Accuracy: 14.71 %"
```

```
# Create Confusion Matrix
conf_matrix <- table(Predicted = multi_log_pred, Actual = train_lc$theme_as_factor)
# Compute Accuracy per Theme (Diagonal / Row Sum)
theme_accuracies <- diag(conf_matrix) / rowSums(conf_matrix)
theme_accuracies <- round(theme_accuracies * 100, 2)  # Convert to percentages
# Print Theme-Wise Accuracy
print(theme_accuracies)
```

```
##   age/power dynamic              crush        empowerment               exes
##              33.33              19.48              31.58               0.00
##     forbidden love              grief         growing up             growth
##               0.00                NaN               6.67               0.00
##              happy               hate         heartbreak           jealousy
##               0.00               0.00              23.08               0.00
##               love      mental health          moving on           partying
##              16.25               7.69               9.88              11.11
##          rebellion           religion        reminiscing            revenge
##               8.70                NaN              26.67               0.00
##      situationship toxic relationship         unrequited
##              20.00               0.00              13.64
```

```
# Calculate column sums
word_sums <- colSums(train_lc[, grepl("^word_", colnames(train_lc))])

# Select columns where sum is at least 2
selected_words <- names(word_sums[word_sums >= 35])

# Subset dataset with only selected word features + target variable
train_lc_filtered <- train_lc[, c("theme_as_factor", selected_words)]
```

```
#full_model <- multinom(theme_as_factor ~ ., data = train_lc_filtered, trace = FALSE)
#step_model <- stepAIC(full_model, direction = "backward", trace = TRUE)

#summary(step_model)  # View selected variables
```