What do Iron Maiden, Taylor Swift and Pharrell Williams have in common? A primer on text mining.*

Walter Guillioli

April 03, 2025

Introduction

I love music and data science. Purpose of this paper is to apply main text analytics techniques to explore lyrics from different authors.

We will see things like sentiment analysis, word clouds, topic model and network bigrams.

Purpose is to process lyrics and get idea of what they sign without reading lyrics.

A key companino for this was the book tidy text oby x (add link)

Data and Methods

Lyrics from 4 albums were dowloaded from the azlyrics.com website. They were manually copy pasted to text documents and read into R. Two albums for Iron Maiden - Number of the Beast (1982) and X-Factor (1995). For Taylor Swift, 1989 from 2014. And Pharrell Williams album x from x.

Data was read from txt into data frame. The techniques applied where: list all of them.

48 songs were analyzed from these albums: Iron Maiden: Number of the Beast 9,
Iron Maiden: X Factor 11, Pharrell Williams: GIRL 12, Taylor Swift: 1989 16.

Table 1 shows the names of the songs of each album.

```
## # A tibble: 48 x 6
##    artist      album               artist_album            song_num song  lyrics
##    <chr>       <chr>               <chr>                   <chr>    <chr> <chr>
##  1 Iron Maiden Number of the Beast Iron Maiden: Number of~ 01       Inva~ "Long~
##  2 Iron Maiden Number of the Beast Iron Maiden: Number of~ 02       Chil~ "He's~
##  3 Iron Maiden Number of the Beast Iron Maiden: Number of~ 03       The ~ "We w~
##  4 Iron Maiden Number of the Beast Iron Maiden: Number of~ 04       22 A~ "If y~
##  5 Iron Maiden Number of the Beast Iron Maiden: Number of~ 05       The ~ "Woe ~
##  6 Iron Maiden Number of the Beast Iron Maiden: Number of~ 06       Run ~ "Whit~
##  7 Iron Maiden Number of the Beast Iron Maiden: Number of~ 07       Gang~ "Shad~
##  8 Iron Maiden Number of the Beast Iron Maiden: Number of~ 08       Hall~ "I'm ~
##  9 Iron Maiden Number of the Beast Iron Maiden: Number of~ 09       Tota~ "Cold~
## 10 Iron Maiden X Factor            Iron Maiden: X Factor   01       Sign~ "Elev~
## # i 38 more rows
```

*The data and scripsts posted here: https://github.com/wguillioli

The lyrics were split into invidual words for analysis and common words (stop words) like a, on, the were removed. Table 2 shows number of words for analysis:

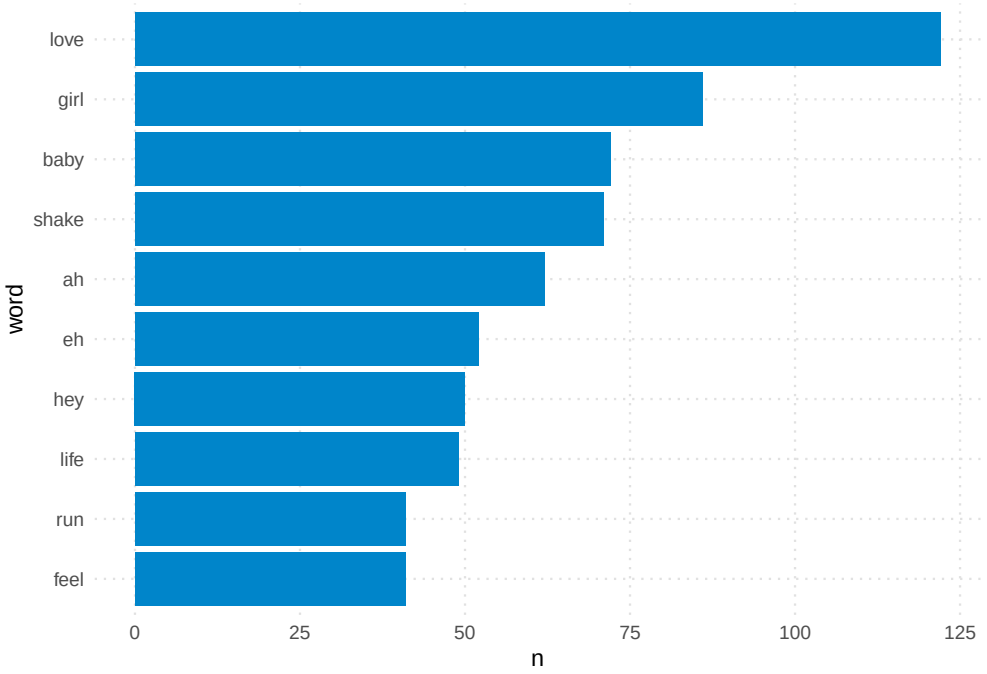
```
## # A tibble: 4 x 2
##   artist_album                      number_of_words
##   <chr>                                       <int>
## 1 Iron Maiden: Number of the Beast              800
## 2 Iron Maiden: X Factor                         871
## 3 Pharrell Williams: GIRL                      1437
## 4 Taylor Swift: 1989                           2000
```

Results

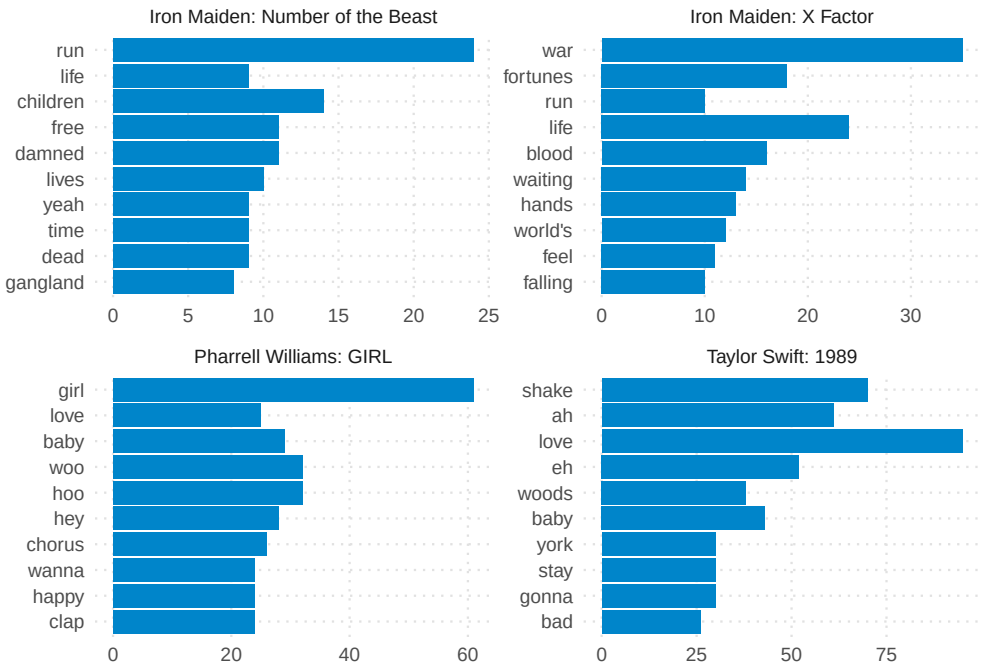
To analyze text we start with x, then do y and then z.

Top words used on the lyrics

We can start by plotting the top 10 most common words across all albums.



This is relatively interesting but not useful since we know some artists use more words than others. So let's do the same but by album.

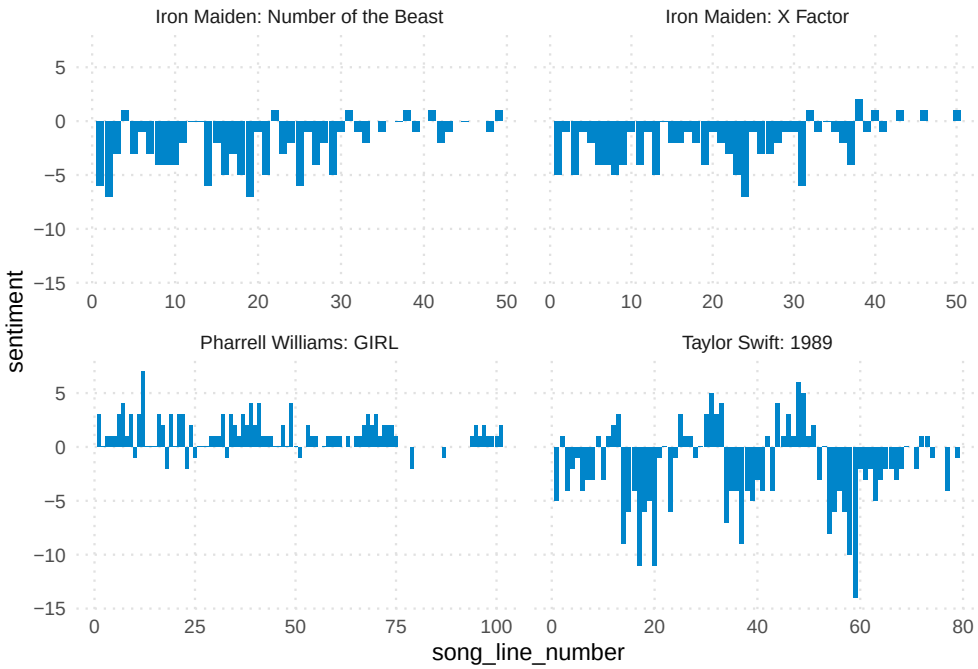


Sentiment

The top words above give an idea of what they talk about but it's just the start. The next thing is to see if they sing about happy, sad or angry stuff. Probably a bit of everything.

We start with a simple analysis to see if the sentiment is positive or negative. In the chart below we see the sentiment evolution of the albums as the album progress where each bar represents a line in the song. Also note that number of lines varies and Maiden sings with less words.

```
## Joining with `by = join_by(word)`
```



Several points to highlight from this chart. Iron Maiden tends to be very negative while Taylor Swift is mostly negative with some exceptions. Pharrell Williams on the contrary seems to be a very happy and uplifting singer. We will explore why and more in the next section.

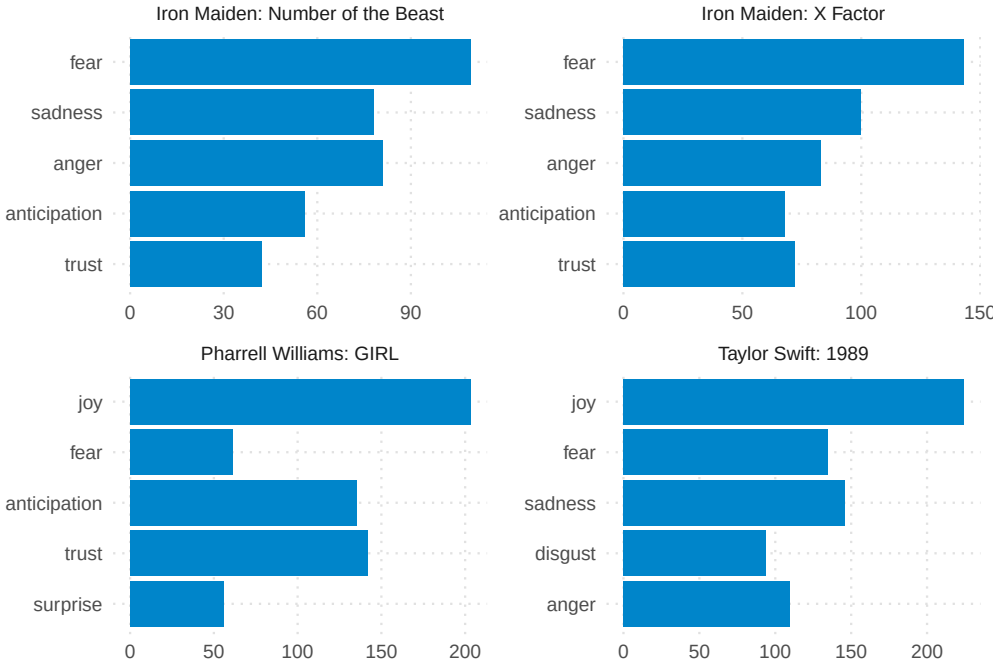
Another thing we can do is to see the sentiment for each album. Maiden is motly about fear, sadness and anger. While Pharrell sings about Joy and Trust. While Taylor seems to be happier but sometimes is inundated with sadness and fear. As we will see later this is a diffrent fear than the one Maiden shows.

```
## Joining with `by = join_by(word)`
```

```
## # A tibble: 20 x 3
## # Groups:   artist_album [4]
##    artist_album                     sentiment        n
##    <chr>                            <chr>        <int>
##  1 Iron Maiden: Number of the Beast fear           109
##  2 Iron Maiden: Number of the Beast anger           81
##  3 Iron Maiden: Number of the Beast sadness         78
##  4 Iron Maiden: Number of the Beast anticipation    56
##  5 Iron Maiden: Number of the Beast trust           42
##  6 Iron Maiden: X Factor            fear           143
##  7 Iron Maiden: X Factor            sadness        100
##  8 Iron Maiden: X Factor            anger           83
##  9 Iron Maiden: X Factor            trust           72
## 10 Iron Maiden: X Factor            anticipation    68
## 11 Pharrell Williams: GIRL          joy            203
## 12 Pharrell Williams: GIRL          trust          142
## 13 Pharrell Williams: GIRL          anticipation   135
## 14 Pharrell Williams: GIRL          fear            61
## 15 Pharrell Williams: GIRL          surprise        56
## 16 Taylor Swift: 1989               joy            224
## 17 Taylor Swift: 1989               sadness        146
```

```
## 18 Taylor Swift: 1989              fear           135
## 19 Taylor Swift: 1989              anger          110
## 20 Taylor Swift: 1989              disgust         94
```

```
## Joining with `by = join_by(word)`
```



sadness is common, so what are the top sad words for each album

nrc_sad <- get_sentiments("nrc") %>% filter(sentiment == "sadness")

lyrics_words %>% #filter(album == "Number of the Beast") %>% inner_join(nrc_sad) %>% count(artist_album, word, sort = TRUE) %>% group_by(artist_album) %>% slice_max(order_by = n, n = 5)

most common and positive words per album

albums_to_parse <- unique(lyrics_words$artist_album)

for (i in 1:length(albums_to_parse)){ print(i)

lyrics_words_top <- lyrics_words %>% filter(artist_album == albums_to_parse[i]) %>% inner_join(bing) %>% count(word, sentiment, sort = TRUE) %>% ungroup()

myplot <- lyrics_words_top %>% group_by(sentiment) %>% slice_max(n, n = 10, with_ties = FALSE) %>% ungroup() %>% mutate(word = reorder(word, n)) %>% ggplot(aes(n, word, fill = sentiment)) + geom_col(show.legend = FALSE) + facet_wrap(~sentiment, scales = "free_y") + labs(x = "Contribution to sentiment", y = NULL) + ggtitle(albums_to_parse[i])

print(myplot)

}

hallowed not positive necesariamente

word clouds by album

set.seed(666) lyrics_words %>% filter(artist_album == albums_to_parse[1]) %>% count(word) %>% with(wordcloud(word, n, max.words = 100))

set.seed(666) lyrics_words %>% filter(artist_album == albums_to_parse[2]) %>% count(word) %>% with(wordcloud(word, n, max.words = 100))

set.seed(666) lyrics_words %>% filter(artist_album == albums_to_parse[3]) %>% count(word) %>% with(wordcloud(word, n, max.words = 100))

plot / calculate most negative songs per album

words_per_song <- lyrics_words %>% group_by(artist_album, song) %>% summarise(words = n())

negative <- bing %>% filter(sentiment == "negative")

negative_words_per_song <- lyrics_words %>% semi_join(negative)

negative_ratio <- negative_words_per_song %>% group_by(artist_album, song) %>% summarise(negativewords =n()) %>% left_join(words_per_song, by = c("artist_album", "song")) %>% mutate(ratio = round(negativewords/words, 2)) %>% ungroup()

ggplot(negative_ratio, aes(song, ratio, fill = artist_album)) + geom_col(show.legend = FALSE) + facet_wrap(~artist_album, ncol = 2, scales = "free_y") + coord_flip()

tf-idf

album_words <- lyrics_df %>% unnest_tokens(word, lyrics) %>% count(artist_album, word, sort = TRUE)

total_words <- album_words %>% group_by(artist_album) %>% summarize(total = sum(n))

album_words <- left_join(album_words, total_words)

album_tf_idf <- album_words %>% bind_tf_idf(word, artist_album, n)

album_tf_idf %>% select(-total) %>% arrange(desc(tf_idf))

album_tf_idf %>% group_by(artist_album) %>% slice_max(tf_idf, n = 15) %>% ungroup() %>% ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = artist_album)) + geom_col(show.legend = FALSE) + facet_wrap(~artist_album, ncol = 2, scales = "free") + labs(x = "tf-idf", y = NULL)

4 n-grams

lyrics_df$lyrics_clean <- removeWords(lyrics_df$lyrics, stop_words$word)

try here with clean lyrics vs normal

lyrics_bigrams <- lyrics_df %>% unnest_tokens(bigram, lyrics, token = “ngrams”, n = 2) %>% filter(!is.na(bigram))

lyrics_bigrams %>% count(bigram, sort = TRUE)

bigrams_separated <- lyrics_bigrams %>% separate(bigram,c(“word1”, “word2”), sep = ” “)

bigrams_filtered <- bigrams_separated %>% filter(!word1 %in% stop_words$word) filter(!word2 %in% stop_words$word)

bigrams_filtered %>% count(word1, word2, sort = TRUE)

bigrams_united <- bigrams_filtered %>% unite(bigram, word1, word2, sep = ” “)

bigrams_tfidf <- bigrams_united %>% count(artist_album, bigram) %>% bind_tf_idf(bigram, artist_album, n) %>% arrange(desc(tf_idf))

bigrams_tfidf %>% group_by(artist_album) %>% slice_max(tf_idf, n = 15) %>% ungroup() %>% ggplot(aes(tf_idf, fct_reorder(bigram, tf_idf), fill = artist_album)) + geom_col(show.legend = FALSE) + facet_wrap(~artist_album, ncol = 2, scales = “free”) + labs(x = “tf-idf”, y = NULL)

function that takes album name of album and ngram filter

and plots the network graph by album

plot_graph <- function(album_to_plot, num){

bigram_counts <- bigrams_filtered %>% filter(artist_album == album_to_plot) %>% count(word1, word2, sort = TRUE)

#bigram_counts

bigram_graph <- bigram_counts %>% filter(n > num) %>% graph_from_data_frame()

a <- grid::arrow(type = “closed”, length = unit(.15, “inches”))

set.seed(97702) ggraph(bigram_graph, layout = “fr”) + geom_edge_link() + geom_node_point(color = “gray”, size = 3) + geom_node_text(aes(label = name), vjust = 1, hjust = 1) + theme_void()

}

plot_graph(albums_to_parse[1], 1)

plot_graph(albums_to_parse[2], 1)

plot_graph(albums_to_parse[3], 2)

plot_graph(albums_to_parse[4], 3)

impact of negated words

negation_words <- c(“not”, “no”, “never”, “without”)

negated_words <- bigrams_separated %>% filter(word1 %in% negation_words) %>% inner_join(afinn, by = c(word2 = “word”)) %>% count(artist_album, word1, word2, value, sort = TRUE)

negated_words %>% mutate(contribution = n * value) %>% arrange(desc(abs(contribution))) %>% head(20) %>% mutate(word2 = reorder(word2, contribution)) %>% ggplot(aes(n * value, word2, fill = n * value > 0)) + geom_col(show.legend = FALSE) + labs(x = “Sentiment value * number of occurrences”, y = “Words preceded by "not"”)

4.2 corr

correlation of words

correlating among albumns

words_pairs <- lyrics_words %>% pairwise_count(word, artist_album, sort = TRUE)

words_pairs table(words_pairs$n)

word_corrs <- lyrics_words %>% group_by(word) %>% filter(n() >= 10) %>% pairwise_cor(word, artist_album, sort = TRUE)

word_corrs

word_corrs %>% filter(item1 %in% c("heaven", "love", "hate", "mad")) %>% group_by(item1) %>% slice_max(correlation, n = 6) %>% ungroup() %>% mutate(item2 = reorder(item2, correlation)) %>% ggplot(aes(item2, correlation)) + geom_bar(stat = "identity") + facet_wrap(~ item1, scales = "free") + coord_flip()

didn't work

LDA topic model on albums

word_counts_beast <- lyrics_words %>% #filter(artist_album == "Iron Maiden: Number of the Beast") %>% count(artist_album, word, sort = TRUE)

beast_dtm <- word_counts_beast %>% cast_dtm(artist_album, word, n)

beast_dtm

lda <- LDA(beast_dtm, k = 4, control = list(seed = 1234)) lda

topics <- tidy(lda, matrix = "beta")

topics

top_terms <- topics %>% group_by(topic) %>% slice_max(beta, n = 10) %>% ungroup() %>% arrange(topic, -beta)

top_terms %>% mutate(term = reorder_within(term, beta, topic)) %>% ggplot(aes(beta, term, fill = factor(topic))) + geom_col(show.legend = FALSE) + facet_wrap(~ topic, scales = "free") + scale_y_reordered()

Conclusions

References

Appendix

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

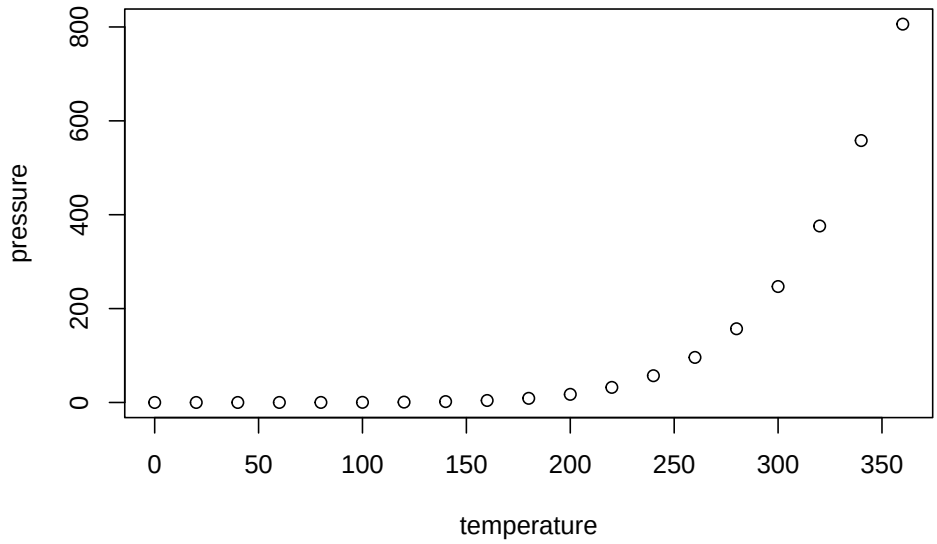
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.