The post [Decode Lyrics in Pop Music: Visualise Prose with the Songsim algorithm](https://lucidmanager.org/decoding-lyrics-with-songsim/) appeared first on [The Lucid Manager](https://lucidmanager.org/).

Music is an inherently mathematical form of art. Ancient Greek mathematician Pythagoras was the first to describe the logic of the scales that form melody and harmony. Numbers can also represent the rhythm of the music. Even the lyrics have a mathematical structure. Poets structure syllables and repeat words to create pleasing sounding prose. This article shows how to decode lyrics from pop songs and visualise them using the Songsim method to analyse their metre.

**Decode Lyrics using the Songsim algorithm**

Data visualiser, pop music appreciator and machine learner [Colin Morris](https://twitter.com/HalfEatenScone) has extensively analysed the repetitiveness of song lyrics. Colin demonstrated that lyrics are becoming more repetitive since the early days of pop music. The most repetitive song is *Around the World* by Daft Punk, which should not be a surprise since the artist repeats the same phrase 144 times. *Bohemian Rhapsody* by Queen has some of the least repetitive lyrics in popular music.

The TedX presentation (see below) by Colin Morris shows how he visualises the repetitiveness of song lyrics with what he calls the [Songsim](https://colinmorris.github.io/SongSim/#/) algorithm. The more points in the graph, the more often a word is repeated.

Visualise the lyrics of Around the World and Bohemian Rhapsody.

**The visual language of song lyrics**

Morris decided to use a self-similarity matrix, used to visualise DNA sequences, to decode lyrics. In this method, the individual words of the song are the names of the columns and the names of the rows in a matrix. For every point in the song where the row name equals the column name, shows a dot. By definition, the diagonal of every similarity matrix is filled. The timeline of the song runs along the diagonal from top left to bottom right.

Patterns away from the diagonal represent two different points in time that have the same words. The more of these patterns we see, the more repetitive a song is. Let’s demonstrate this with the [first words ever recorded](https://www.youtube.com/watch?v=g3qPT30LejM) by Thomas Edison in 1877.

Mary had a little lamb, whose fleece was white as snow. And everywhere that Mary went, the lamb was sure to go.

The similarity matrix below visualises the two first sentences of the famous nursery rhyme. It shows where the words “Mary”, “lamb” and “was” are repeated once.

Self-similarity matrix for Mary had a Little Lamb by Thomas Edison.

The snowflake diagrams are a visual language to decode lyrics. The verses are the gutters with only diagonal lines. A verse is not very repetitive besides some stop words. The verse repeats through the song. Many songs have a bridge that contrasts with the rest of the song. The bridge is in most songs a unique pattern with self-similarity.

The diagram below visualises the lyrics of one of the most famous pop songs ever, *Waterloo* by Abba. The first 30 words are the opening verse, which shows little repetition, other than stop words such as and the pronoun I. After that we see diagonal lines appearing that represent the repetitive use of the song title. Towards the end of the song, we see the bridge, which is like a little snowflake within the diagram.

Decoding lyrics: Waterloo by Abba.

The next section shows how to implement this approach with ggplot, scraping pop song lyrics from the azlyrics.com website.

**Implementing Songsim with ggplot**

The code below visualises song lyrics or poetry as suggested by Colin Morris. The code uses four libraries. I use the tidyverse series of libraries because it makes life very easy. The tidytext library uses the tidyverse principles to analyse text. The old reshape2 library helps to transform a matrix, and lastly, rvest helps to scrape song lyrics from the azlyrics website.

The first function scrapes song lyrics from the azlyrics website using the artist and song as input. The first three lines clean the artist and song variables. This code removes any character that is not a number or a letter, converts to lowercase and lastly removes the definite article in the artist name. These two fields are then concatenated to create the URL, which the function prints. The remainder of the code scrapes the lyrics from the website or trips on an error 404 when it cannot find the song/artist combination.

The second function implements the Morris method to visualise the lyrics. The code extracts single words from the text and places them in a data frame (tibble). This data frame is converted to a boolean matrix that contains the visualisation.

The code looks at each word and places the value TRUE where reappears in the song. Each of the vectors is then concatenated to a matrix. Lastly, ggplot visualises the matrix is visualised as a raster.

What does your favourite song look like a snowflake diagram?

**The Songsim code**

You can view the code below or download the latest version from my [GitHub repository](https://github.com/pprevos/r.prevos.net/blob/master/Miscellaneous/lyrics.R).

## Decoding lyrics

library(tidyverse)

library(tidytext)

library(reshape2)

library(rvest)

get\_lyrics <- function(artist, song) {

artist <- gsub("[^A-Za-z0-9]+", "", tolower(artist))

song <- gsub("[^A-Za-z0-9]+", "", tolower(song))

artist <- gsub("^the", "", artist)

url = paste("http://azlyrics.com/lyrics/",

artist, "/", song, ".html", sep = "")

print(url)

azlyrics <- read\_html(url)

lyrics <- html\_nodes(azlyrics, "div")

lyrics <- html\_text(lyrics[22])

gsub("\r|\n", " ", lyrics)

}

plot\_snowflake <- function(artist, song){

lyrics <- get\_lyrics(artist, song)

lyrics <- data\_frame(line = lyrics) %>%

filter(line != "")

words <- lyrics %>%

unnest\_tokens(word, line)

words\_matrix <- lapply(1:nrow(words),

function(w){

as.character(words[w, 1]) == words

}

) %>%

do.call(cbind, .)

rownames(words\_matrix) <- 1:nrow(words)

colnames(words\_matrix) <- 1:nrow(words)

melt(words\_matrix, varnames = c("x", "y")) %>%

ggplot(aes(x, -y, fill = value)) +

geom\_raster() +

scale\_fill\_manual(values = c("white", "dodgerblue4"), guide = FALSE) +

theme\_void() +

ggtitle(artist, subtitle = song)

}

plot\_snowflake("Abba", "Waterloo")