There are two packages – geniusR and geniusr – which will do this. I played with both and found geniusR easier to use. Neither is perfect, but what is perfect, anyway?

Now you’ll want to load geniusR and tidyverse so we can work with our data.

library(geniusR)  
library(tidyverse)

## ── Attaching packages ────────────────────────────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──

##  ggplot2 2.2.1  purrr 0.2.4  
##  tibble 1.4.2  dplyr 0.7.4  
##  tidyr 0.8.0  stringr 1.3.0  
##  readr 1.1.1  forcats 0.3.0

## ── Conflicts ───────────────────────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
##  dplyr::filter() masks stats::filter()  
##  dplyr::lag() masks stats::lag()

For today’s demonstration, I’ll be working with data from two artists I love: Taylor Swift and Lorde. Both dropped new albums last year, Reputation and Melodrama, respectively, and both, though similar in age and friends with each other, have very different writing and musical styles.

geniusR has a function genius\_album that will download lyrics from an entire album, labeling it by track.

swift\_lyrics <- genius\_album(artist="Taylor Swift", album="Reputation")

## Joining, by = c("track\_title", "track\_n", "track\_url")

lorde\_lyrics <- genius\_album(artist="Lorde", album="Melodrama")

## Joining, by = c("track\_title", "track\_n", "track\_url")

Now we want to tokenize our datasets, remove stop words, and count word frequency – this code should look familiar, except this time, I’m combining them using the pipeline symbol (%>%) from the tidyverse, which allows you to string together multiple functions without having to nest them.

library(tidytext)  
tidy\_swift <- swift\_lyrics %>%  
 unnest\_tokens(word,lyric) %>%  
 anti\_join(stop\_words) %>%  
 count(word, sort=TRUE)

## Joining, by = "word"

head(tidy\_swift)

## # A tibble: 6 x 2  
## word n  
##   
## 1 call 46  
## 2 wanna 37  
## 3 ooh 35  
## 4 ha 34  
## 5 ah 33  
## 6 time 32

tidy\_lorde <- lorde\_lyrics %>%  
 unnest\_tokens(word,lyric) %>%  
 anti\_join(stop\_words) %>%  
 count(word, sort=TRUE)

## Joining, by = "word"

head(tidy\_lorde)

## # A tibble: 6 x 2  
## word n  
##   
## 1 boom 40  
## 2 love 26  
## 3 shit 24  
## 4 dynamite 22  
## 5 homemade 22  
## 6 light 22

Looking at the top 6 words for each, it doesn’t look like there will be a lot of overlap. But let’s explore that, shall we? Lorde’s album is 3 tracks shorter than Taylor Swift’s. To make sure our word comparisons are meaningful, I’ll create new variables that takes into account total number of words, so each word metric will be a proportion, allowing for direct comparisons. And because I’ll be joining the datasets, I’ll be sure to label these new columns by artist name.

tidy\_swift <- tidy\_swift %>%  
 rename(swift\_n = n) %>%  
 mutate(swift\_prop = swift\_n/sum(swift\_n))  
  
tidy\_lorde <- tidy\_lorde %>%  
 rename(lorde\_n = n) %>%  
 mutate(lorde\_prop = lorde\_n/sum(lorde\_n))

There are multiple types of joins available in the tidyverse. I used an anti\_join to remove stop words. Today, I want to use a full\_join, because I want my final dataset to retain all words from both artists. When one dataset contributes a word not found in the other artist’s set, it will fill those variables in with missing values.

compare\_words <- tidy\_swift %>%  
 full\_join(tidy\_lorde, by = "word")  
  
summary(compare\_words)

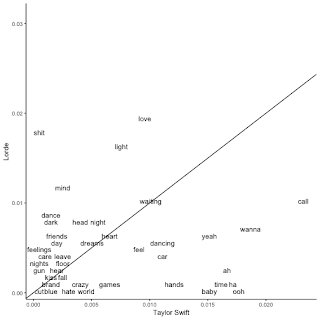
## word swift\_n swift\_prop lorde\_n   
## Length:957 Min. : 1.000 Min. :0.00050 Min. : 1.0   
## Class :character 1st Qu.: 1.000 1st Qu.:0.00050 1st Qu.: 1.0   
## Mode :character Median : 1.000 Median :0.00050 Median : 1.0   
## Mean : 3.021 Mean :0.00152 Mean : 2.9   
## 3rd Qu.: 3.000 3rd Qu.:0.00151 3rd Qu.: 3.0   
## Max. :46.000 Max. :0.02321 Max. :40.0   
## NA's :301 NA's :301 NA's :508   
## lorde\_prop   
## Min. :0.0008   
## 1st Qu.:0.0008   
## Median :0.0008   
## Mean :0.0022   
## 3rd Qu.:0.0023   
## Max. :0.0307   
## NA's :508

The final dataset contains 957 tokens – unique words – and the NAs tell how many words are only present in one artist’s corpus. Lorde uses 301 words Taylor Swift does not, and Taylor Swift uses 508 words that Lorde does not. That leaves 148 words on which they overlap.

There are many things we could do with these data, but let’s visualize words and proportions, with one artist on the x-axis and the other on the y-axis.

ggplot(compare\_words, aes(x=swift\_prop, y=lorde\_prop)) +  
 geom\_abline() +  
 geom\_text(aes(label=word), check\_overlap=TRUE, vjust=1.5) +  
 labs(y="Lorde", x="Taylor Swift") + theme\_classic()

## Warning: Removed 809 rows containing missing values (geom\_text).

[](https://i1.wp.com/4.bp.blogspot.com/-nRnN6fYkO78/WvZcS5ugnbI/AAAAAAAALic/YDNRzq54QSEewUwUJ6KzgC83r-zNEuacgCLcBGAs/s1600/unnamed-chunk-7-1.png?ssl=1)

The warning lets me know there are 809 rows with missing values – those are the words only present in one artist’s corpus. Words that fall on or near the line are used at similar rates between artists. Words above the line are used more by Lorde than Taylor Swift, and words below the line are used more by Taylor Swift than Lorde. This tells us that, for instance, Lorde uses “love,” “light,” and, yes, “shit,” more than Swift, while Swift uses “call,” “wanna,” and “hands” more than Lorde. They use words like “waiting,” “heart,” and “dreams” at similar rates. Rates are low overall, but if you look at the max values for the proportion variables, Swift’s most common word only accounts for about 2.3% of her total words; Lorde’s most common word only accounts for about 3.1% of her total words.

This highlights why it’s important to remove stop words for these types of analyses; otherwise, our datasets and chart would be full of words like “the,” “a”, and “and.”