Text analysis

Jason G. Fleischer, Ph.D.
Asst. Teaching Professor
Department of Cognitive Science, UC San Diego
jfleischer@ucsd.edu

@jasongfleischer

https://jgfleischer.com

Examples of questions that require text analysis

1. Did J.K. Rowling write The Cuckoo's Calling under the pen name Robert Galbraith?
2. What themes are common in 19th century literature?
3. Can we tell the difference between tweets that come from Trump himself or a staffer?
4. Is Hillary the most poisoned name in US History?
5. Can we visualize the narrative structure of The Hobbit?
6. Who has the biggest vocabulary in hip hop?
7. Is there a gender imbalance in who gets to say lines in the movies?

A whole bunch of text processing project ideas

- Spam detection (in text messages, on social media)
- Hate speech detection
- Predictive text (like gmail)
- Summarize articles (TL;DR)
- Summarize trends on social media… or in cutting edge NLP research?
- Plagiarism detection… in code projects?
- Document similarity… prevent the posting of duplicate questions on Campuswire?

Today's example question: How has pop music changed in the last five years?

Goal: Understand the basics of sentiment analysis and TF-IDF

What data would we need to answer this question?

How has pop music changed in the last five years?

Data: Lyrics to the most popular songs from each year

The data : Top songs from Feb music charts 2017-2021

2017: 152 songs

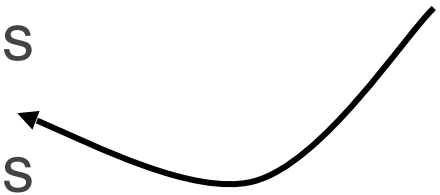
2018: 139 songs

2019: 127 songs

2020: 137 songs

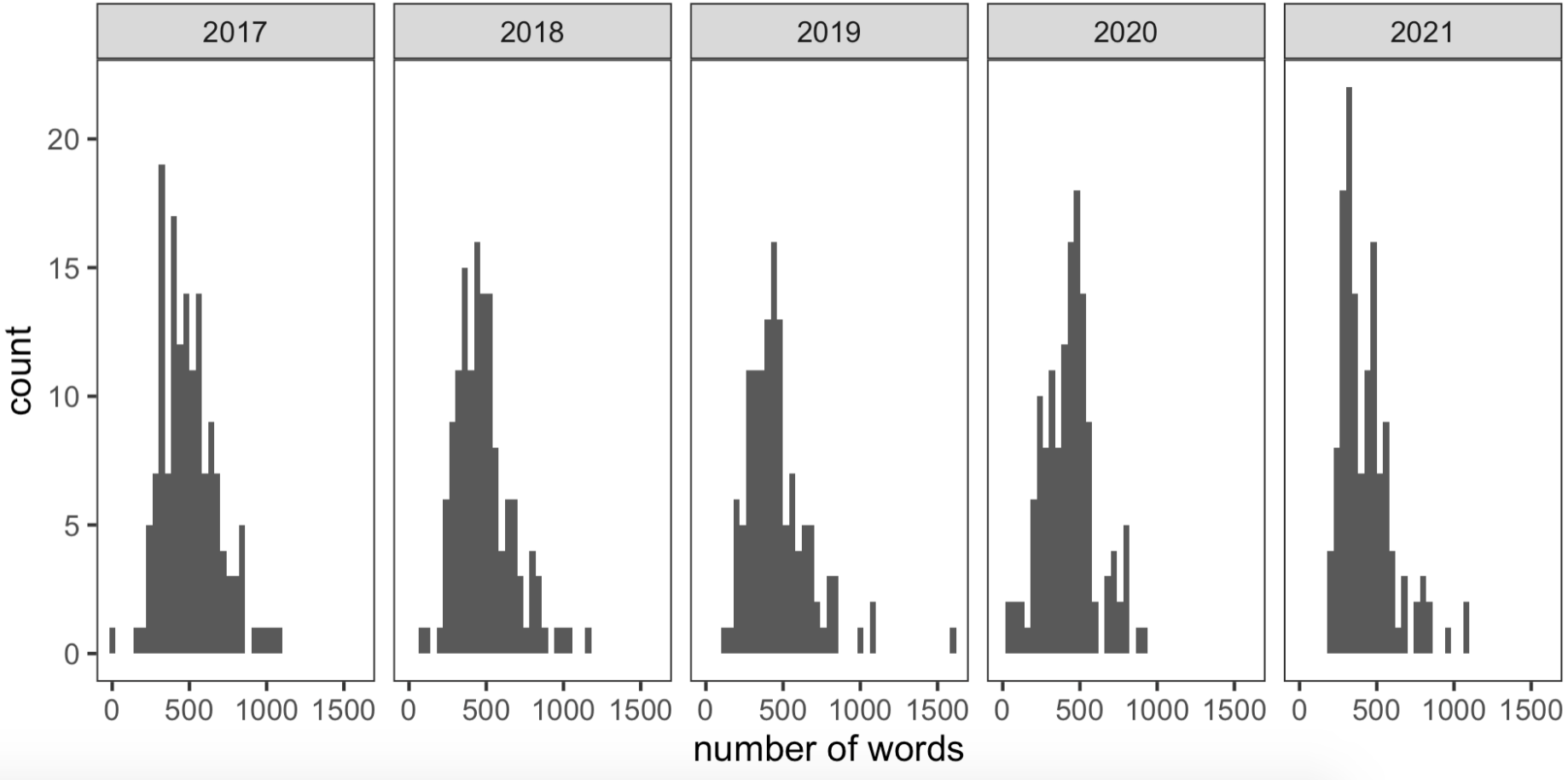
2021: 134 songs

Song data from **Spotify**.
Lyrics from **genius.com**

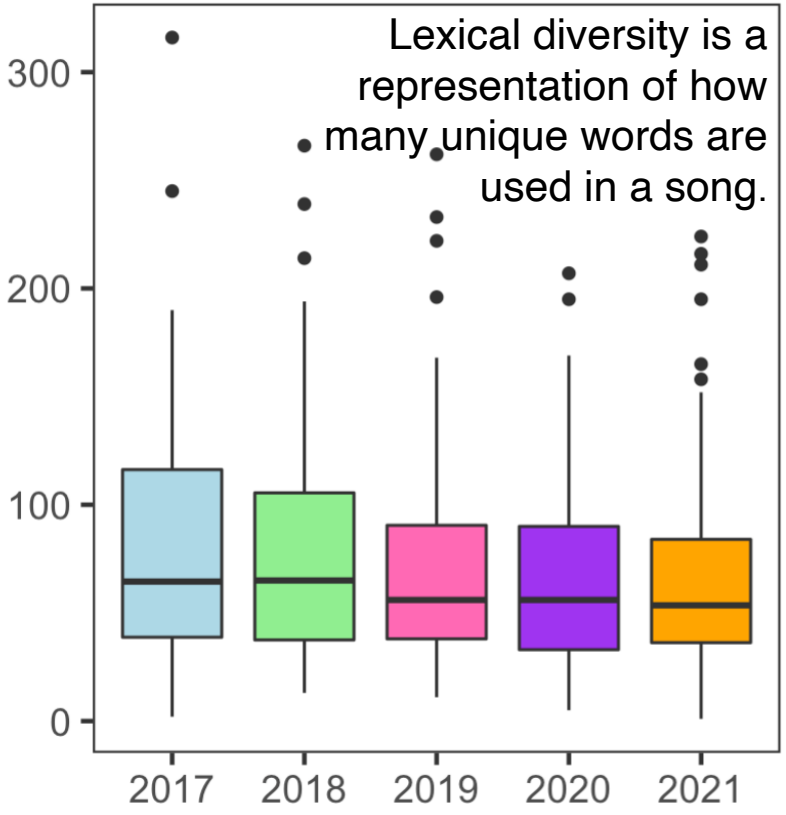


Questions we can ask...

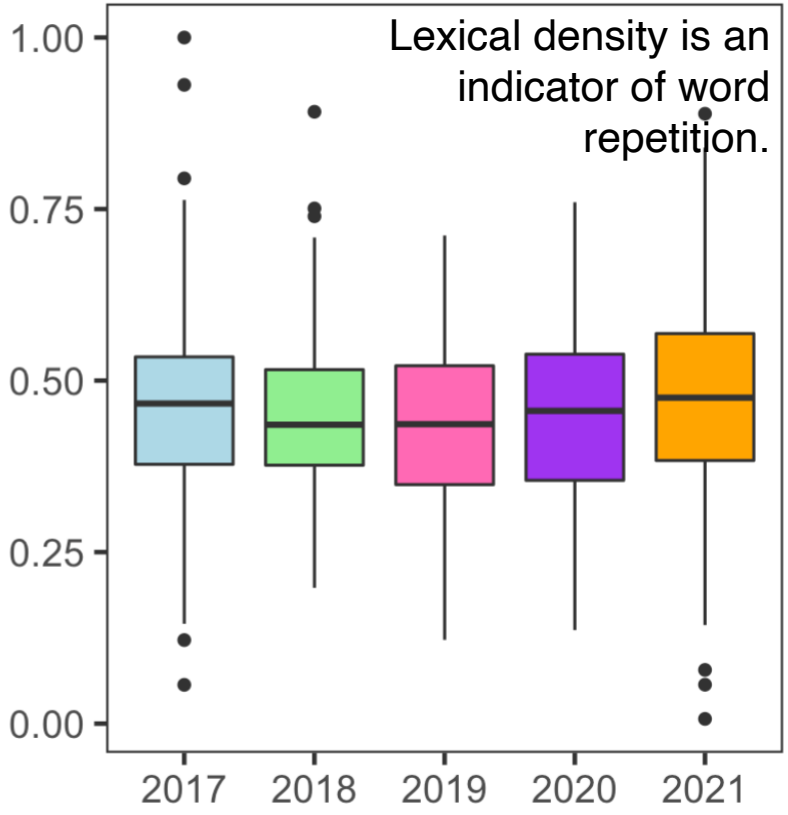
1. Does the total number of words change over time?
2. Does uniqueness change over time?
3. Does the diversity or density change?
4. What words are most common?
5. What words are most unique to each year?
6. What sentiment do songs convey most frequently?
7. Has sentiment changed over time?
8. What are the sentiment of the #1 songs?
9. What words contribute to the sentiment of these #1 songs?
10. ...what about bigrams? N-grams?


2017
2018
2019
2020
2021
count
0
5
10
15
20
0
500
1000
1500
number of words

Lexical Diversity



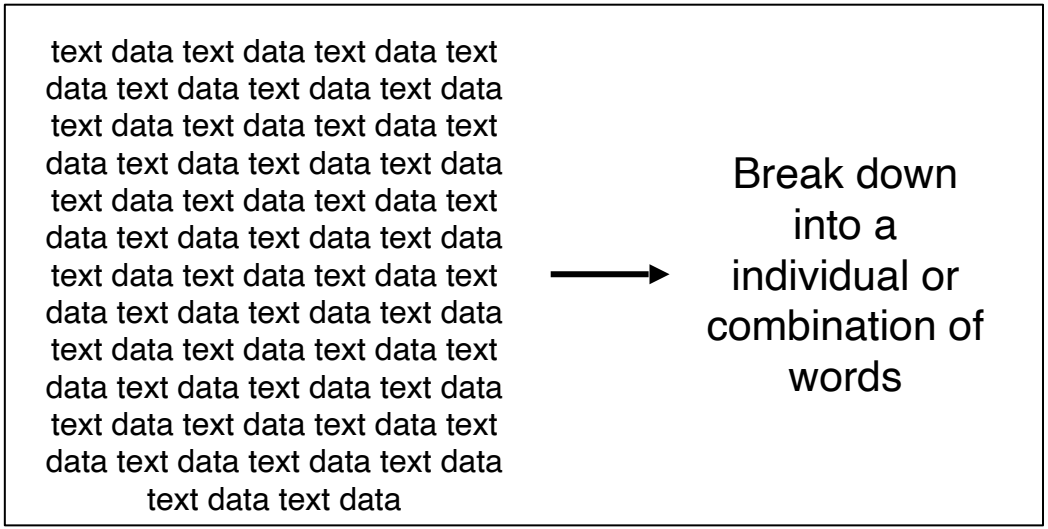
Lexical Density



Sentiment Analysis

Sentiment Analysis

Programmatically infer emotional content of text



Part of the "NRC" sentiment **lexicon**:

word	sentiment	lexicon
<chr>	<chr>	<chr>
abacus	trust	nrc
abandon	fear	nrc
abandon	negative	nrc
abandon	sadness	nrc
abandoned	anger	nrc
abandoned	fear	nrc
abandoned	negative	nrc
abandoned	sadness	nrc
abandonment	anger	nrc
abandonment	fear	nrc

... with 27,304 more rows

When doing sentiment analysis...

token - a meaningful unit of text

- what you use for analysis
- *tokenization* takes corpus of text and splits it into tokens (words, bigrams, etc.)

Check out https://neptune.ai/blog/tokenization-in-nlp

stop words - words not helpful for analysis

- extremely common words such as “a”,“the”, “of”, “to”
- are typically removed from analysis

When doing sentiment analysis...

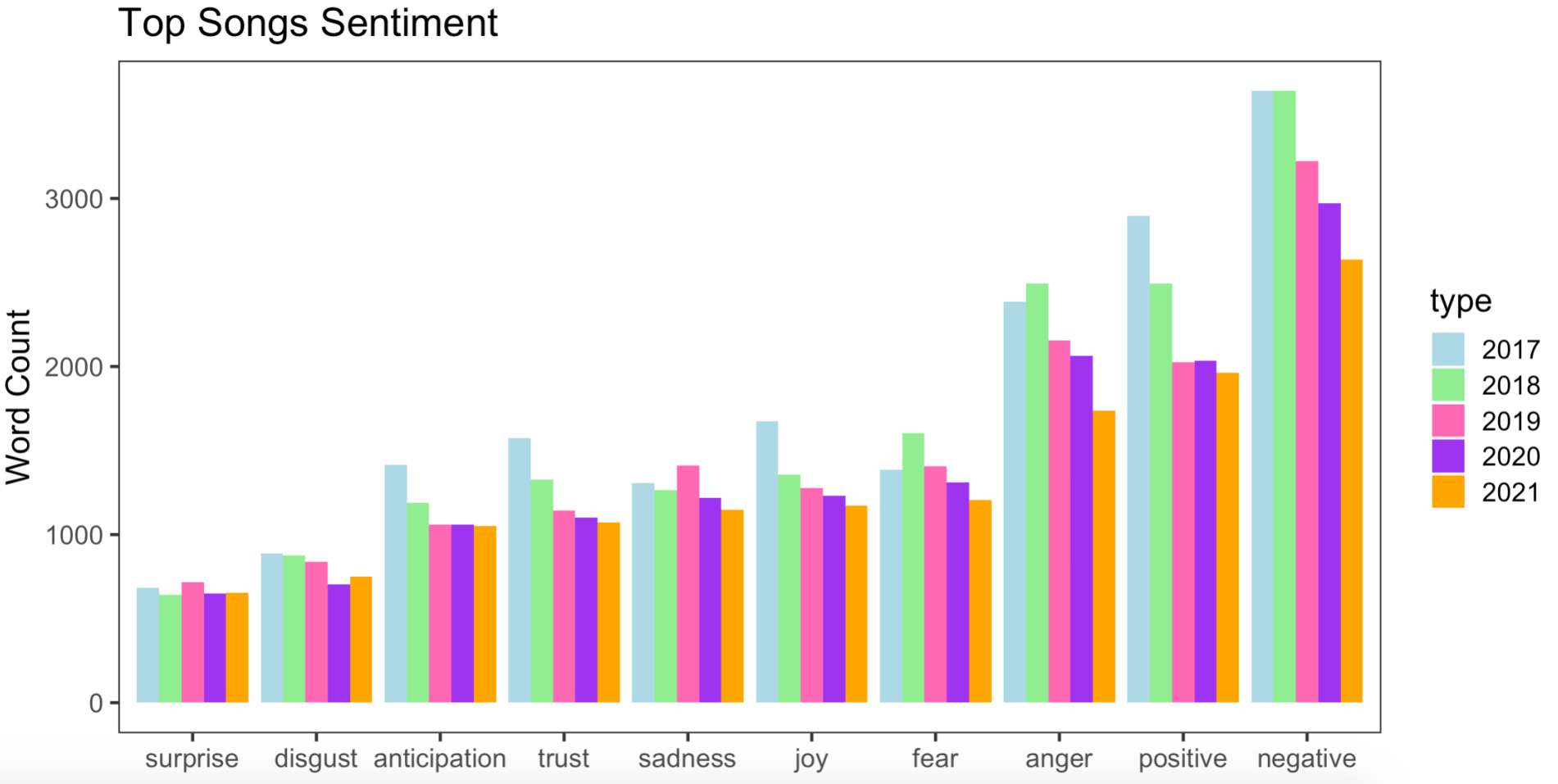
stemming or lemmatization

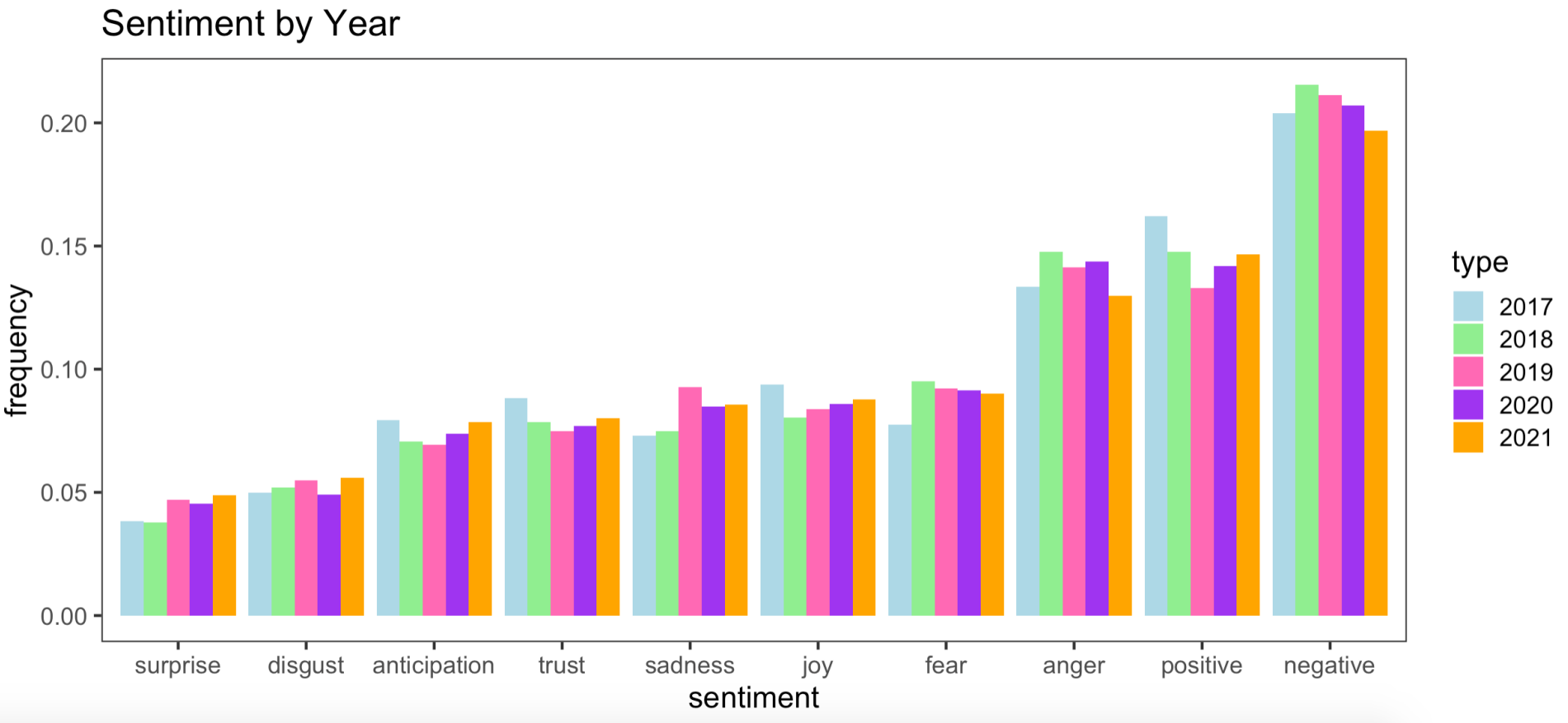
- Identifying the root for each token
- Jumping, jumped, jumps, jump all have the same root ‘jump’
- Where things get tricky: jumper???
- Stemmers and lemmatizers do the same thing, stemmers are word focussed/cruder and lemmatizers try to take into account the context to correctly stem jumper differently depending on if its “I love my new jumper” (In the US jumper=a kind of dress, in the UK jumper=sweater) or its “Shane is a long jumper”

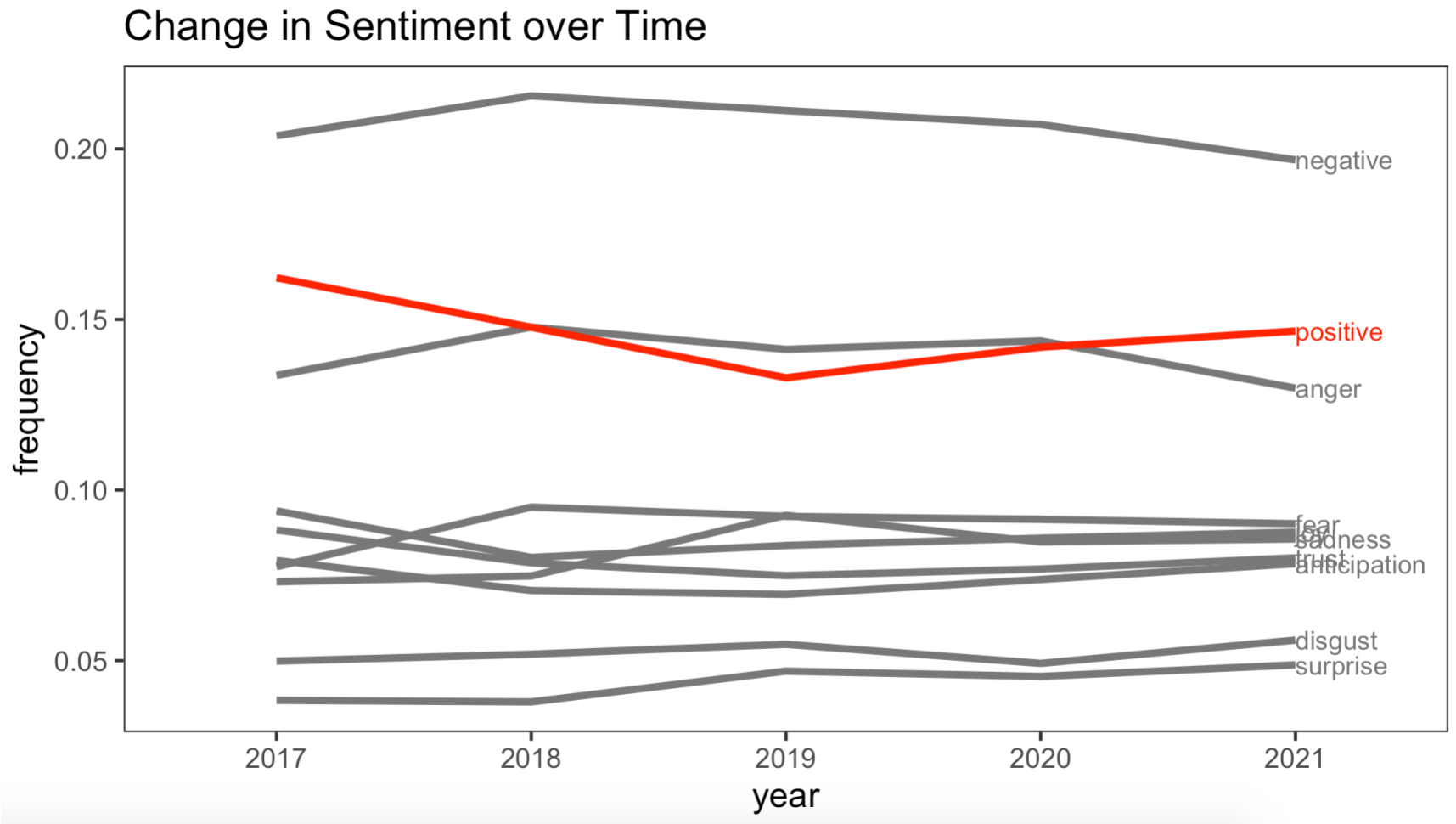
Check out https://www.datacamp.com/tutorial/stemming-lemmatization-python

In text analysis, your choices matter:

1. How to tokenize?
2. Remove stop words? Remove common words?
3. Use stemming/lemmatization?
4. What sentiment lexicon to use?


Top Songs Sentiment
Word Count
3000
2000
1000
0
surprise
disgust
anticipation
trust
sadness
joy
fear
anger
positive
negative
type
2017
2018
2019
2020
2021


Sentiment by Year
frequency
0.20
0.15
0.10
0.05
0.00
surprise
disgust
anticipation
trust
sadness
joy
fear
anger
positive
negative
sentiment
type
2017
2018
2019
2020
2021


Change in Sentiment over Time
frequency
0.20
0.15
0.10
0.05
2017
2018
2019
2020
2021
year
negative
positive
anger
fear
joy
sadness
trust
anticipation
disgust
surprise

Sentiment: Driver's License

0
3
6
9

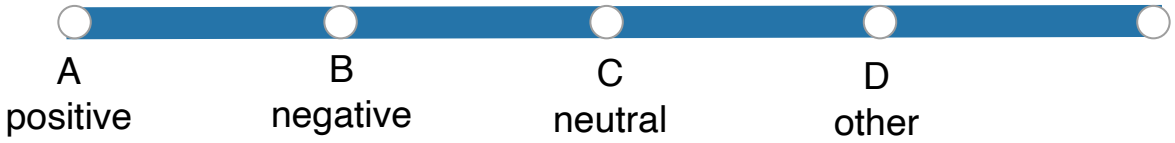
Word Count

I got my driver's license last week
Just like we always talked about
'Cause you were so excited for me
To finally drive up to your house
But today I drove through the suburbs
Crying 'cause you weren't around
And you're probably with that blonde girl
Who always made me doubt
She's so much older than me
She's everything I'm insecure about
Yeah, today I drove through the suburbs
'Cause how could I ever love someone else?
And I know we weren't perfect but I've never felt this way for no one
And I just can't imagine how you could be so okay now that I'm gone
Guess you didn't mean what you wrote in that song about me
'Cause you said forever, now I drive alone past your street
And all my friends are tired
Of hearing how much I miss you, but
I kinda feel sorry for them
'Cause they'll never know you the way that I do, yeah
Today I drove through the suburbs
And pictured I was driving home to you

Sentiment Limitations

How would you classify the sentiment of the following sentence?

“The idea behind the movie was great, but it could have been better”



“Dude!”

“So bad!”

The limitations of sentiment analysis:
Context, homonyms, pragmatics, parts of speech

Mid, Shade

The limitations of sentiment analysis:
Lexicon not up to the task, lexicon specific results

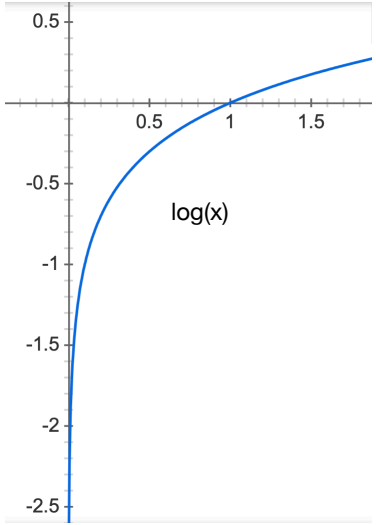
TF-IDF:
Term frequency - Inverse document frequency

Given a corpus of documents, figure out which words are uniquely important to each document

TF-IDF:
Term Frequency - Inverse Document Frequency

Term Frequency (TF) : how frequently a word occurs in a document

Inverse document frequency (IDF) : how important a word is to a document



$$idf(\text{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

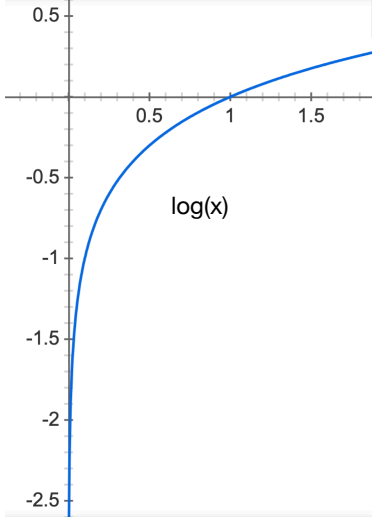
decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents

TF-IDF:

Term Frequency - Inverse Document Frequency

the frequency of a term adjusted for how rarely it is used

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$



TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

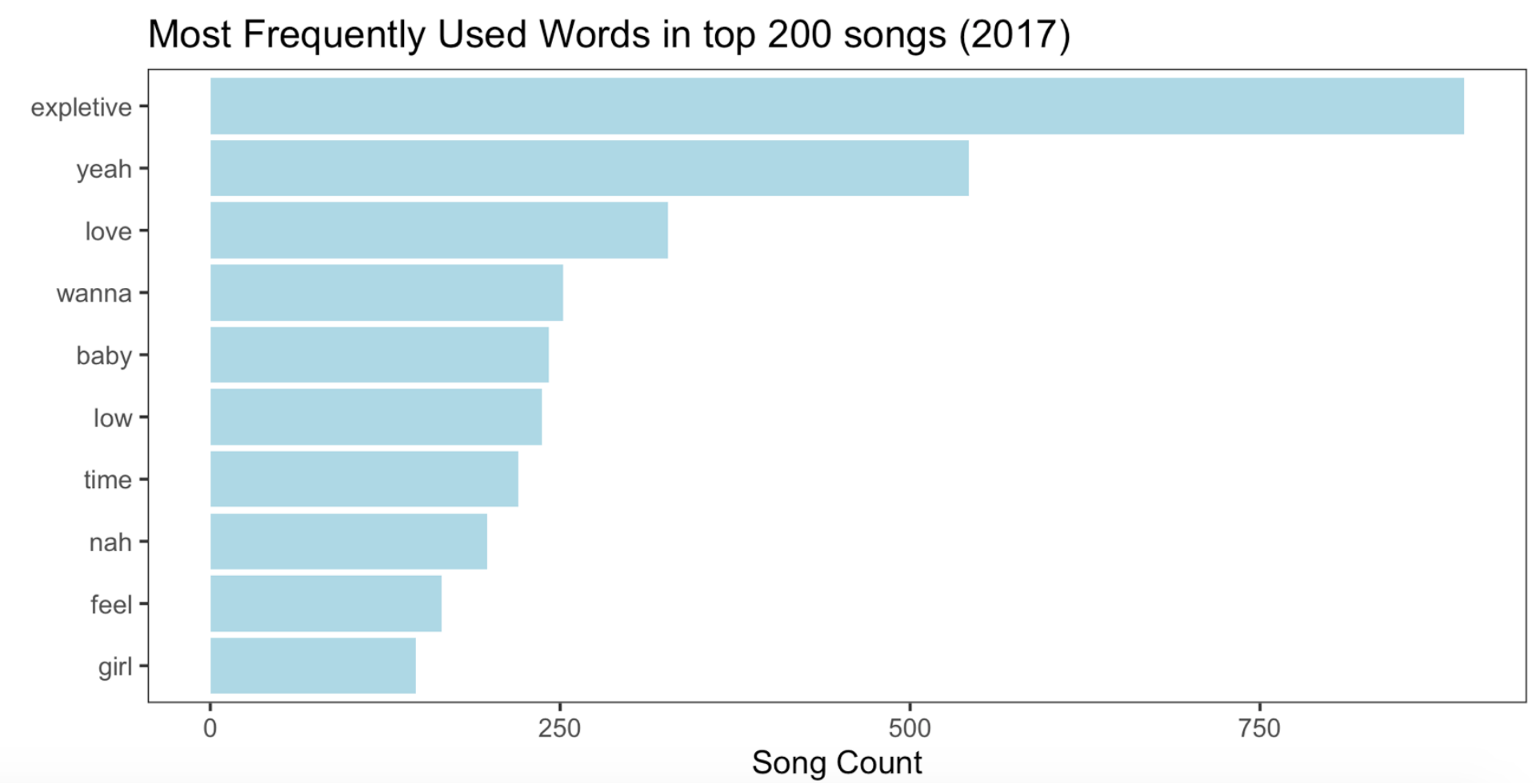
df_x = number of documents containing x

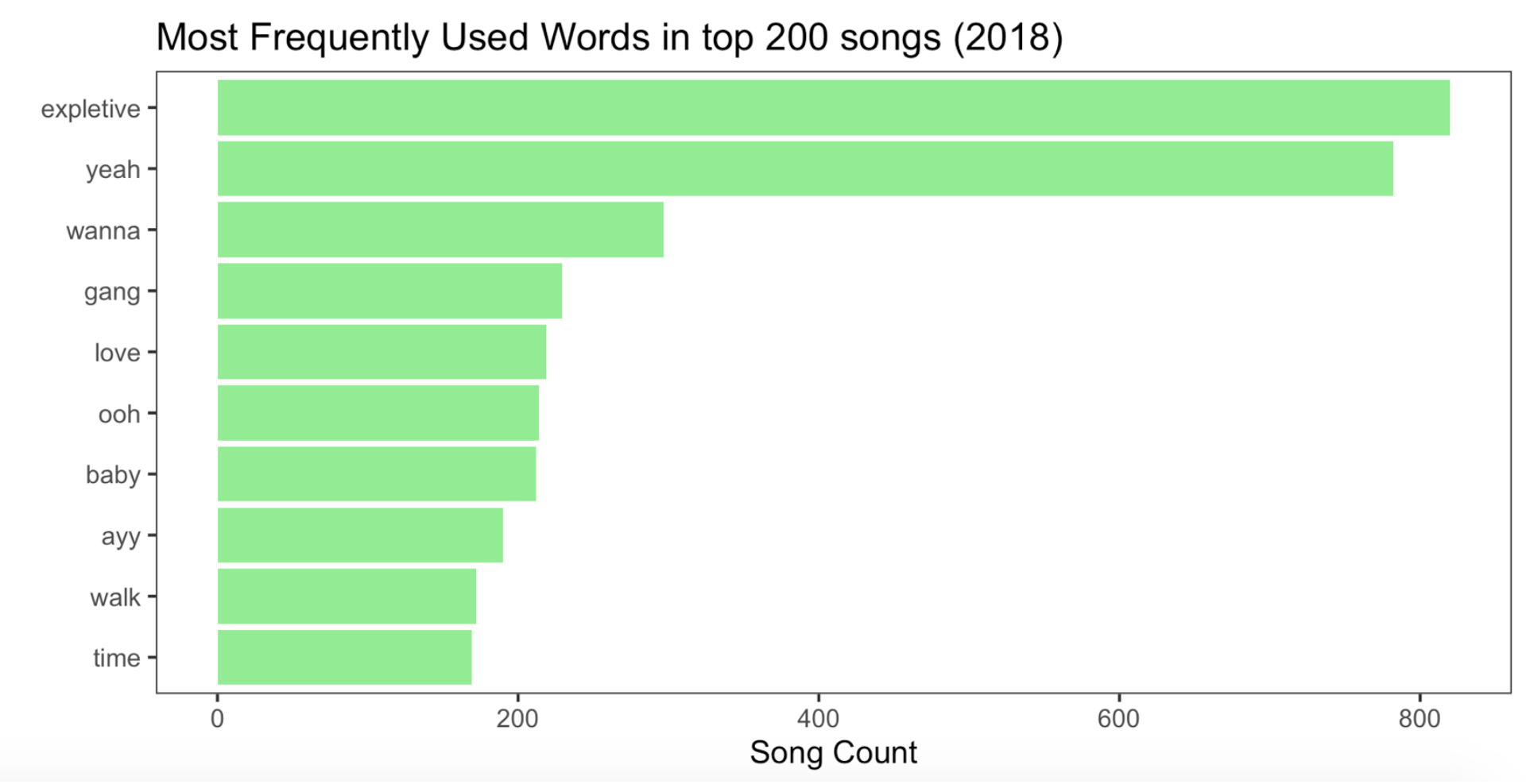
N = total number of documents

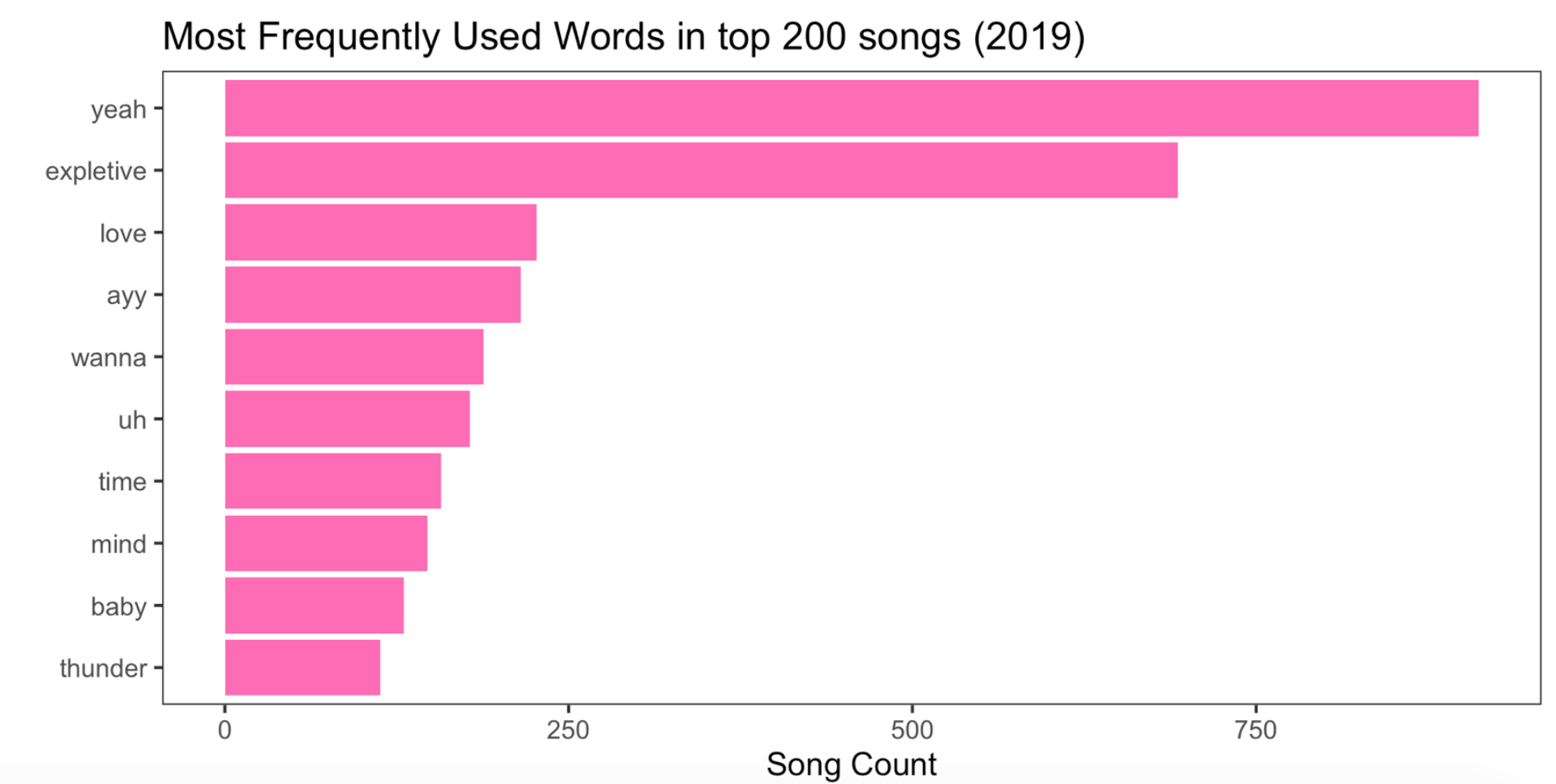
What words are the most unique to the lyrics of each year's top hits?

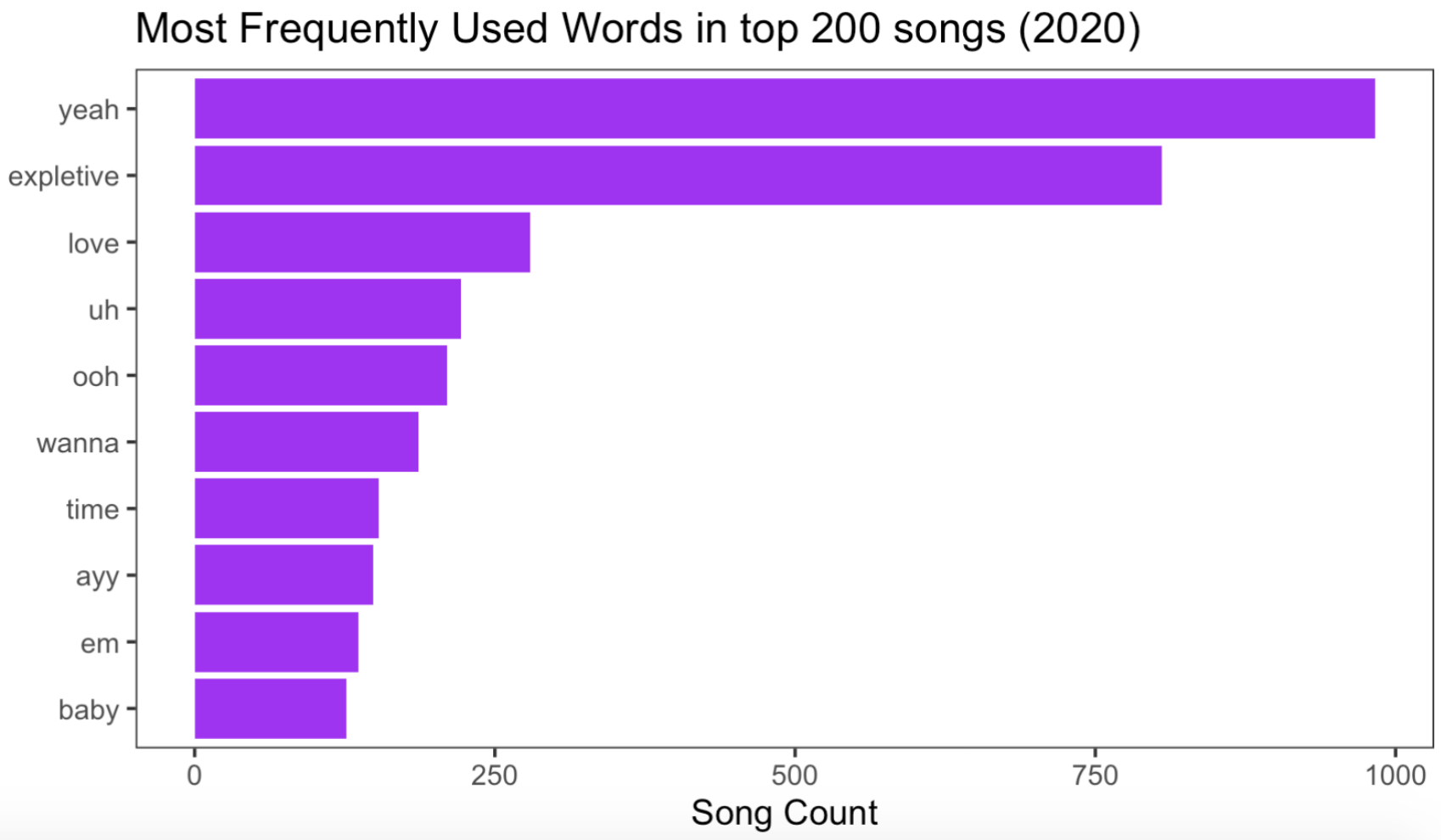
Goal: to use TF-IDF to *find the important words* for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents

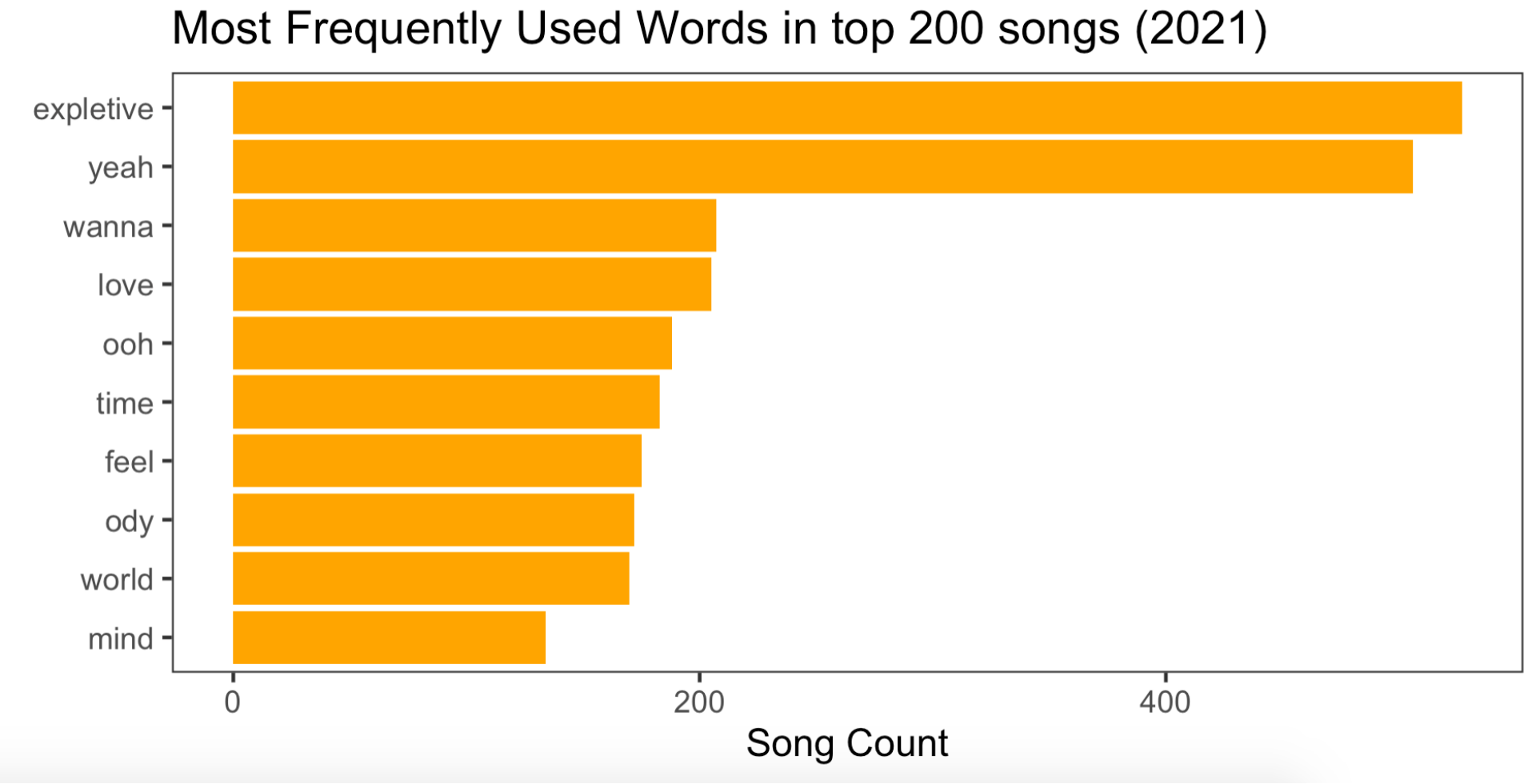
Calculating TF-IDF attempts to find the words that are important (i.e., common) in a text, but not *too* common


Most Frequently Used Words in top 200 songs (2017)
expletive
yeah
love
wanna
baby
low
time
nah
feel
girl
0
250
500
750
Song Count


Most Frequently Used Words in top 200 songs (2018)
expletive
yeah
wanna
gang
love
ooh
baby
ayy
walk
time
0
200
400
600
800
Song Count


Most Frequently Used Words in top 200 songs (2019)
yeah
expletive
love
ayy
wanna
uh
time
mind
baby
thunder
0
250
500
750
Song Count


Most Frequently Used Words in top 200 songs (2020)
yeah
expletive
love
uh
ooh
wanna
time
ayy
em
baby
0
250
500
750
1000
Song Count


Most Frequently Used Words in top 200 songs (2021)
expletive
yeah
wanna
love
ooh
time
feel
ody
world
mind
0
200
400
Song Count

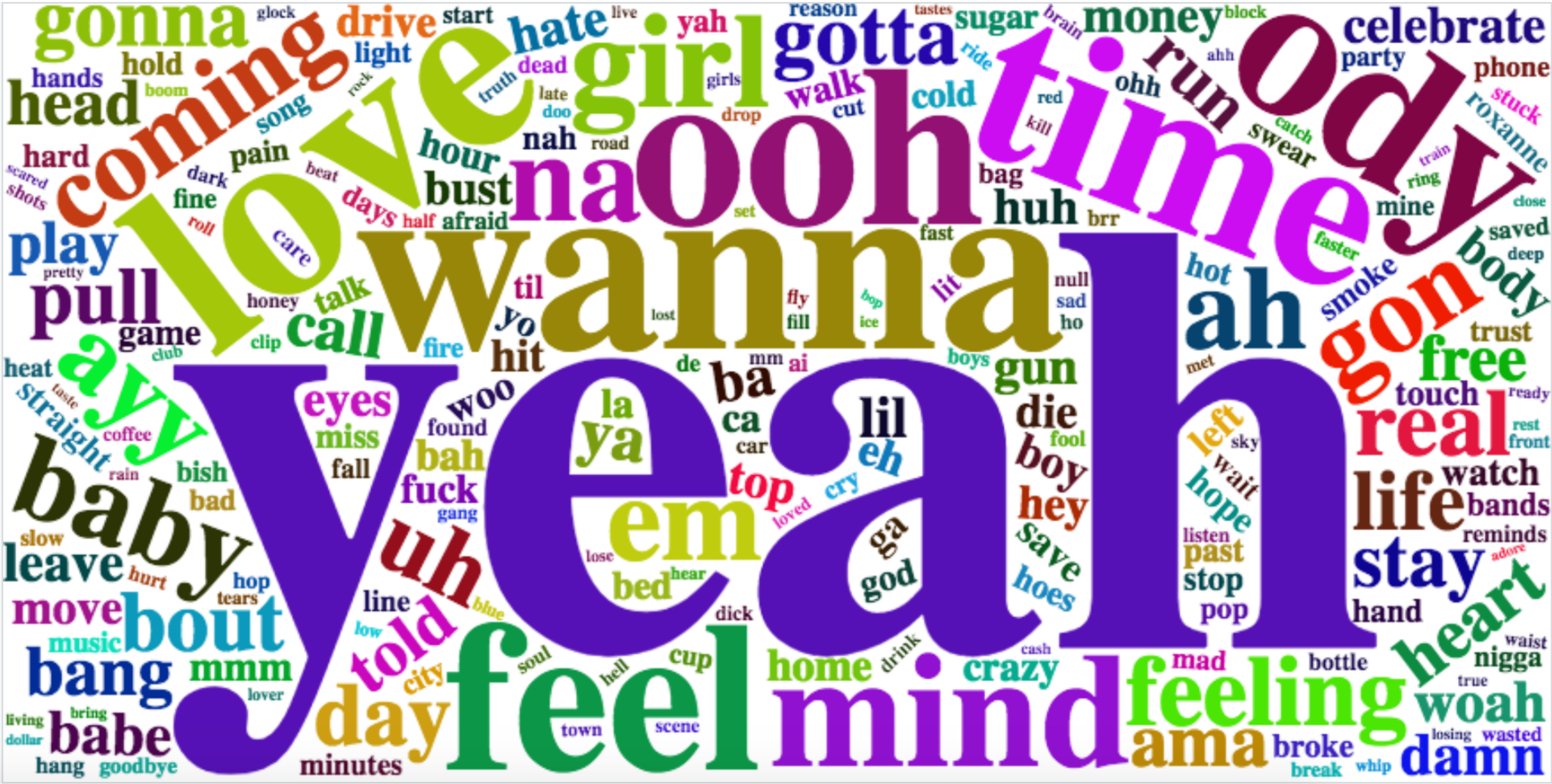
Term Frequency can only tell us so much….

2017

2018

2019

2020


yeah
wanna
ooh
love
time
body
girl
na
gotta
coming
gonna
feel
mind
feeling
ah
gon
real
life
baby
ayy
uh
em
bout
bang
babe
day
told
stay
heart
woah
damn
ama
pull
play
head
hate
drive
money
celebrate
run
free
call
bust
hour
hit
ba
lil
die
boy
hey
top
eyes
miss
bah
fuck
ya
la
ca
eh
ga
god
bed
home
crazy
cup
mmm
move
leave
music
hang
goodbye
minutes
town
scene
broke
break
whip
wasted
losing
bands
reminds
watch
touch
trust
saved
deep
close
ready
rest
front
waist
nigga
true
bottle
mad
hand
pop
stop
past
listen
hope
wait
left
sky
fool
save
hoes
gun
boys
met
smoke
hot
faster
mine
ring
train
roxanne
stuck
phone
party
ohh
ahh
block
swear
catch
brr
huh
bag
red
kill
ride
brain
sugar
tastes
reason
yah
start
live
glock
light
hold
hands
boom
song
rock
truth
dead
late
doo
girls
walk
cut
cold
drop
nah
road
pain
dark
hard
scared
shots
roll
fine
beat
days
half
afraid
care
pretty
talk
honey
game
club
clip
fire
yo
til
lit
fast
set
null
sad
ho
fly
fill
bop
ice
lost
mm
ai
de
woo
found
fall
bish
rain
coffee
taste
straight
heat
slow
bad
hurt
hop
tears
line
blue
low
city
soul
hell
lover
bring
living
dollar
gang
loved
cry
car
lose
hear
dick
drink
cash

2021

TF-IDF:
Term Frequency - Inverse Document Frequency

Term Frequency (TF) : how frequently a word occurs in a document

Inverse document frequency (IDF) : intended to measure how important a word is to a document

decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents

$$idf(\text{term}) = \ln\left(\frac{n_{\text{documents}}}{n_{\text{documents containing term}}}\right)$$

TF-IDF:

Term Frequency - Inverse Document Frequency

the frequency of a term adjusted for how rarely it is used

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

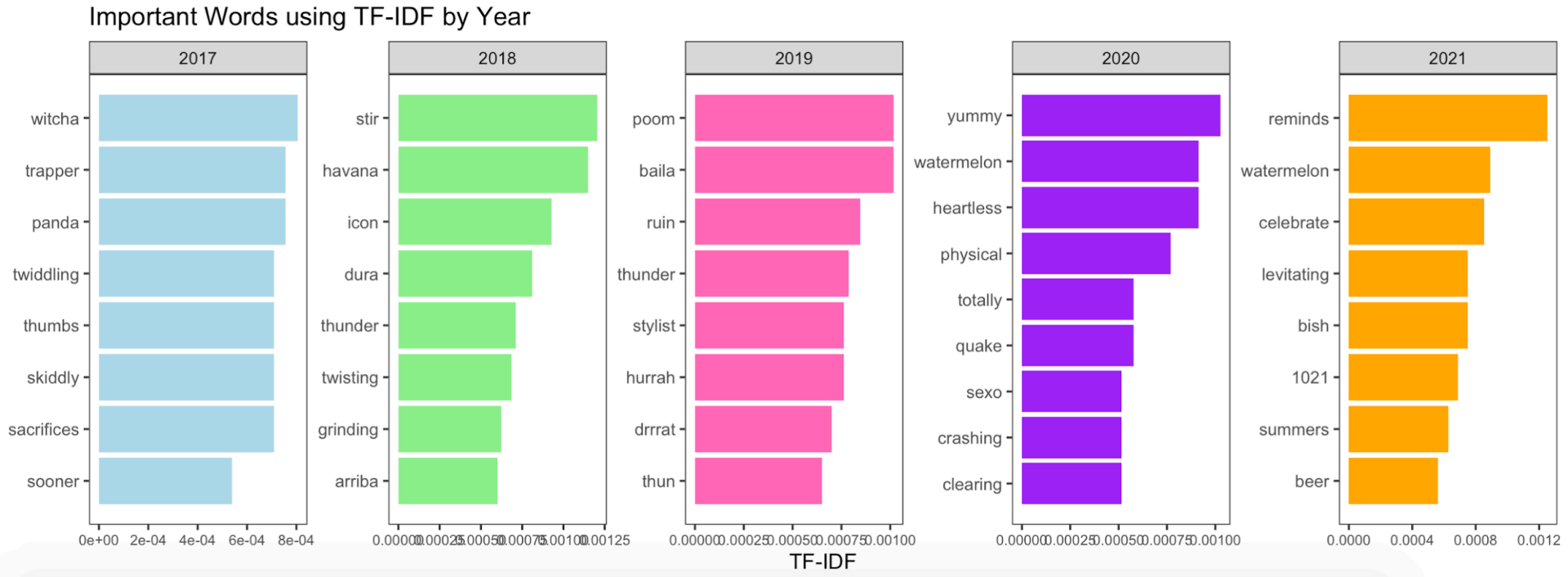
TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents


Important Words using TF-IDF by Year
2017
witcha
trapper
panda
twiddling
thumbs
skiddly
sacrifices
sooner
0e+00
2e-04
4e-04
6e-04
8e-04
2018
stir
havana
icon
dura
thunder
twisting
grinding
arriba
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
2019
poom
baila
ruin
thunder
stylist
hurrah
drrrat
thun
0.00000
0.00025
0.00050
0.00075
0.00100
2020
yummy
watermelon
heartless
physical
totally
quake
sexo
crashing
clearing
0.00000
0.00025
0.00050
0.00075
0.00100
2021
reminds
watermelon
celebrate
levitating
bish
1021
summers
beer
0.0000
0.0004
0.0008
0.0012
TF-IDF

Father and Son
Flight of the Conchords

Live in London
2019

Just you and I
Driving round in the car
I even let you drive
Eating dinner from a can
dad's **yummy** can delight

I'm Me
Chai

Punk
2019

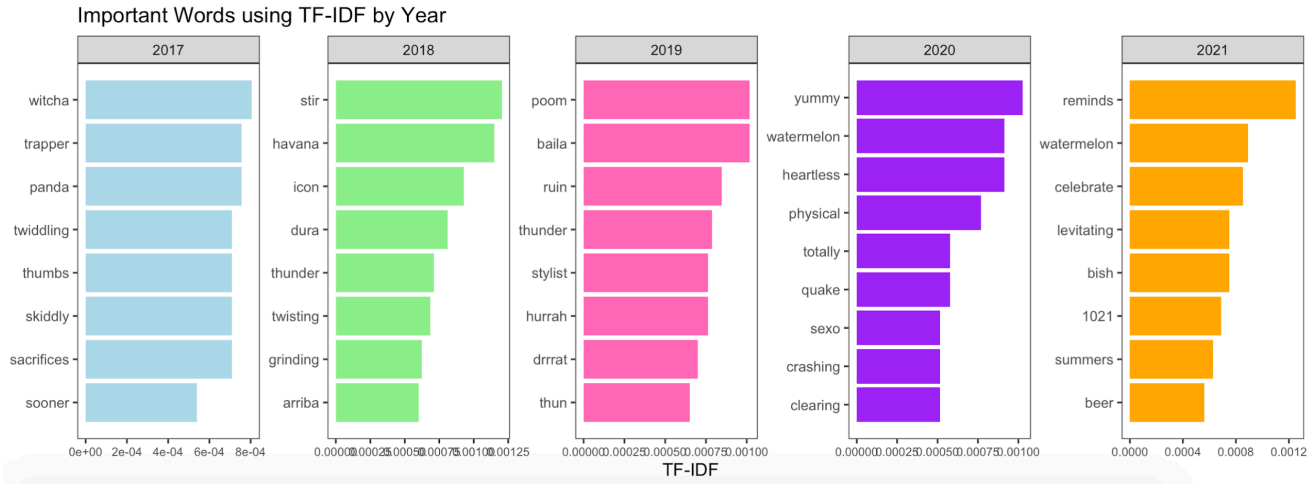
ビーフ アンド チャオズ フィッシュ オレンジ
Everything **yummy** foods!

Sex Talk
Megan Thee Stallion

Sex Talk
2019

He eat from the back to the front
Bitch I got green like I'm Buttercup (ah)
He say I got that lil **yummy** yum (ah ah)

What can you conclude from this TF-IDF plot?



A No words overlap across the years in these data

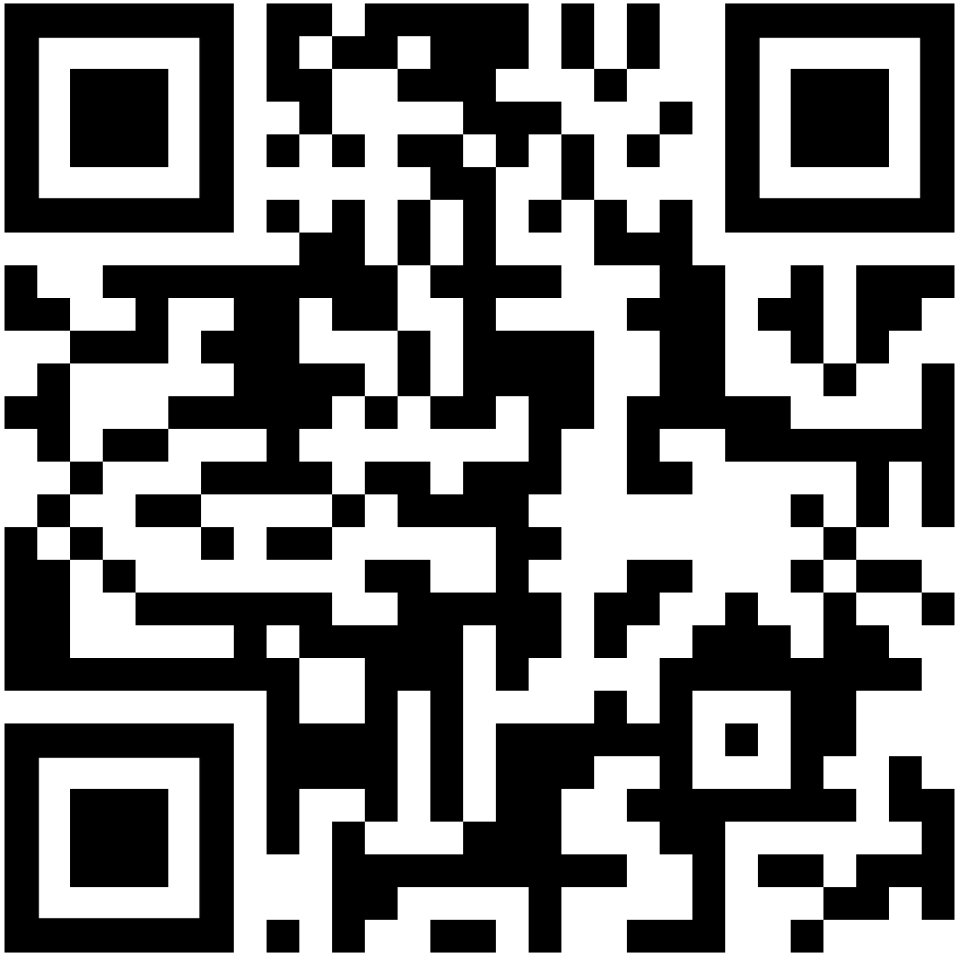
B ‘reminds’ and ‘watermelon’ are the most unique words to the 2021 data

C ‘watermelon’ is the most common word in this dataset

D A-C (all of the above)

E None of the above

https://forms.gle/uD8cVTUUzAsxcVvKA



Questions we can ask...

1. Does the total number of words change over time?
2. Does uniqueness change over time?
3. Does the diversity or density change?
4. What words are most common?
5. What words are most unique to each year?
6. What sentiment do songs convey most frequently?
7. Has sentiment changed over time?
8. What are the sentiment of the #1 songs?
9. What words contribute to the sentiment of these #1 songs?
10. ...what about bigrams? N-grams?

EDA

TF-IDF

Sentiment Analysis

Text analysis

The big picture

- Identify the problem you are trying to solve… what you want to answer suggests the approaches you will use below
- Clean and preprocess the data
- Linguistic transformations
    - Tokenization, lemmatization, stemming, remove stop words
    - [optional] part of speech tagging, named entity recognition
- EDA
- [optional] Calculate measurements and statistics
    - e.g., diversity, density, sentiment, TF-IDF
- [optional] Transform representation
    - Bag of words, vector embeddings
- [optional] Train a ML system or write an algorithm to accomplish a task given the representation
    - e.g., detect spam