A Visual Guide to the Hits of the Ages

Basic Information

Title: A visual guide to the hits of the ages

Sam Himes, uID: u0947447, email: u0947447@utah.edu

John Stanley, uID: u0873475, email: u0873475@utah.edu

GitHub Link: https://github.com/john-stanley2/DataVis-Project

Background and Motivation

Discuss your motivations and reasons for choosing this project, especially any background or research interests that may have influenced your decision.

We both have an appreciation of music and shared curiosity about the evolution of music over time. We also have an interest in how music correlates with human culture. This project provides us an opportunity to communicate and explore music data and show examples of how music may reflect changing human thoughts and preferences over time.

Project Objectives

Provide the primary questions you are trying to answer with your visualization. What would you like to learn and accomplish? List the benefits.

We want to explore some of the specific ways that music has changed over the years. Primarily, we will examine the repetitiveness of songs over time and genre. Additionally, we will quantify and visualize repeated word patterns and themes over time a music genre. We hope to visualize some of the ways that popular music has changed over time. Ideally, we would like to answer questions like, do newer songs repeat themselves more or less than older songs? And when did the word “funky” start getting used in popular music?

Data

From where and how are you collecting your data? If appropriate, provide a link to your data sources

We will be using combining data from the website “Acclaimed music” (http://www.acclaimedmusic.net) and API for “Genius” (https://genius.com/).

“Acclaimed Music” was created by Henrik Franzon from Sweden. His purpose was to “[write] a computer program to determine ‘ultimate’ lists of albums and songs.” His list of acclaimed music uses an aggregation of ranked lists from critics, music artists, radio hosts, etc. to create a single ranked list of the best songs and albums of all time. He shares the statistics used to aggregate the list and admits that there will always be bias in both the statistics and critical opinions in music.

For the purposes of this assignment, we will be using the song/album titles, genre, and year from this dataset. We will be using this data (joined with lyric data) as an unordered sample of songs and albums to create interesting and insightful visualizations.

“Genius” was created in 2009 as an encyclopedia to music, with a particular focus on music lyrics. They provide a publicly accessible API that can search and return music lyrics and other data. You can search by artist, album, or song title. For our project, we will be using song/album titles from “Acclaimed music” as search criteria to gather lyrics.

We plan to join derived data from the returned lyrics with the acclaimed music dataset. Specifics for derived data and working with the lyrics data are described in more detail below.

Data Processing

Do you expect to do substantial data cleanup? What quantities do you plan to derive from your data? How will data processing be implemented?

We will have to do a fair bit of data cleanup to do. Our input data will be the raw csv’s from the acclaimed music website. These csv’s include the name of the song/album, the artist, the year it came out, and some scoring metric for how to rank the song. For our project, we will be ignoring the scoring metrics, and focusing on the other values of the data frame. We will be using the python library lyricsgenius to interact with the genius.com website (the website that collects lyrics for songs). We will use some python commands to get the full lyrics for each song/album and the genre of each song. After we get the full lyrics for each song, we will have to clean the lyrics up. For example, some songs have sound effects that are written in the lyrics, for example, “*indistinct chatter*” or “*Car radio Noises*”. These strings should not count toward the total words in a song. Additionally, we will take other common NLP preprocessing steps like converting everything to lower case and removing non-alphanumeric characters. After the text is processed, we will count the number of words and unique words in each song. After this step, derive two more things from the lyrics. First, we will remove non-stop words from the lyrics. (Stop words include words like “a”, “of”, “or”, and “and”) Then we will get the top 10 most common words for each and their respective frequencies.

Genre information is not readily available on the API or “acclaimed music” csv’s, and may require web scraping the “acclaimed music” website, or further exploring the genius API for ways to extract genre. This data will be joined with the collected lyric data.

Second, we will lemmatize the lyrics and count the frequency of some pre-defined words and topics. We will write all this information to a csv. By the end of our processing, our csv will have the following column names.

Song Name	4th Popular Word Count	10th Popular Word Count	"Fresh" Count
Artist Name	5th Popular Word	Love Count	"Groovy" Count
Number of Words	5th Popular Word Count	Swear Count	"Hip" Count
Number of Unique Words	6th Popular Word	Money Count	"Drunk" Count
Genre	6th Popular Word Count	Dance Count	"Shorty" Count
1st Popular Word	7th Popular Word	Pain Count	"Twerk" Count
1st Popular Word Count	7th Popular Word Count	Family Count	"Selfie" Count
2nd Popular Word	8th Popular Word	Money Count	"Chill" Count
2nd Popular Word Count	8th Popular Word Count	God Count	"Darling" Count
3rd Popular Word	9th Popular Word	"Funky" Count	"Bae" Count
3rd Popular Word Count	9th Popular Word Count	"Cool" Count	"FOMO" Count
4th Popular Word	10th Popular Word	"Rock" Count	Year of Song

The column like "Money Count" (columns without quotes) be looking for mentions of a concept. Words like "money", "cash", "dollars", and "bills" would all contribute to the count. Columns like "'Groovy' Count" would only search for the word "groovy".

Visualization Design

How will you display your data? Provide some general ideas that you have for the visualization design. Develop ***three alternative prototype designs for your visualization****. Create* ***one final design that incorporates the best of your three designs****. Describe your designs and justify your choices of visual encodings. We recommend you use the Five Design Sheet Methodology.*

Prototypes are also provided in the folder along with this document.

Figures 1, 2, and 3 attempt to visualize the change in word uniqueness over time, as well as the distribution of song uniqueness for a given year/decade. We attempt to use position as the primary visual channel, and use familiar graphs (histogram, bar chart, and line chart). This makes the data readily accessible and relatively accurate to read.

We attempted several methods to show the distribution of uniqueness per time period, such as using bars (Figure 1, first graphic), uses separate lines (Figure 2, first graphic), and creating an entirely separate distribution (Figure 1, last graphic).

The buttons/arrows on Figure 1 are meant to represent interactivity for filtering the data based on genre. The vertical line in Figure 2, graphic 1 is meant to represent interactivity for gathering or filtering data based on mouse position.

Figure 1

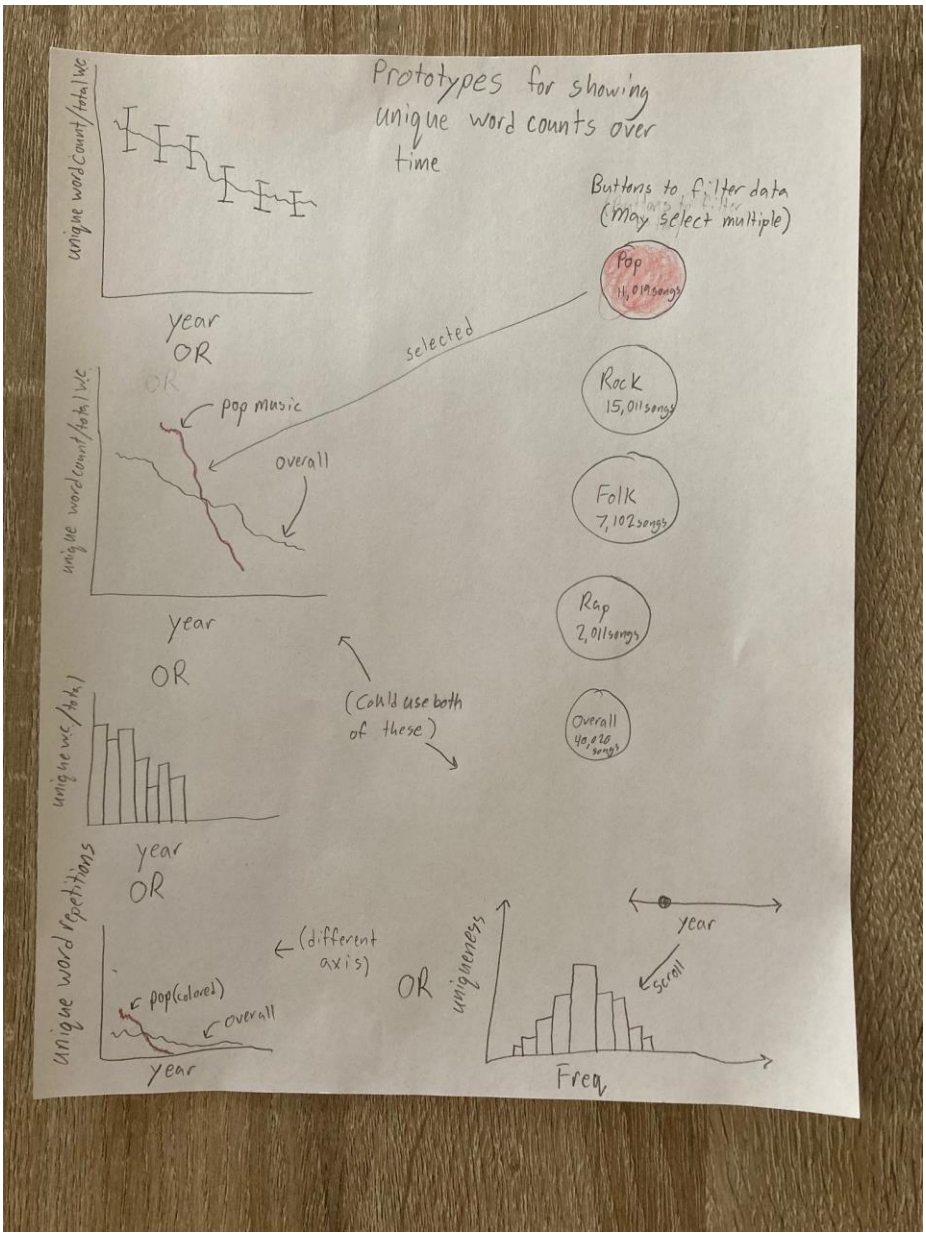


Figure 2

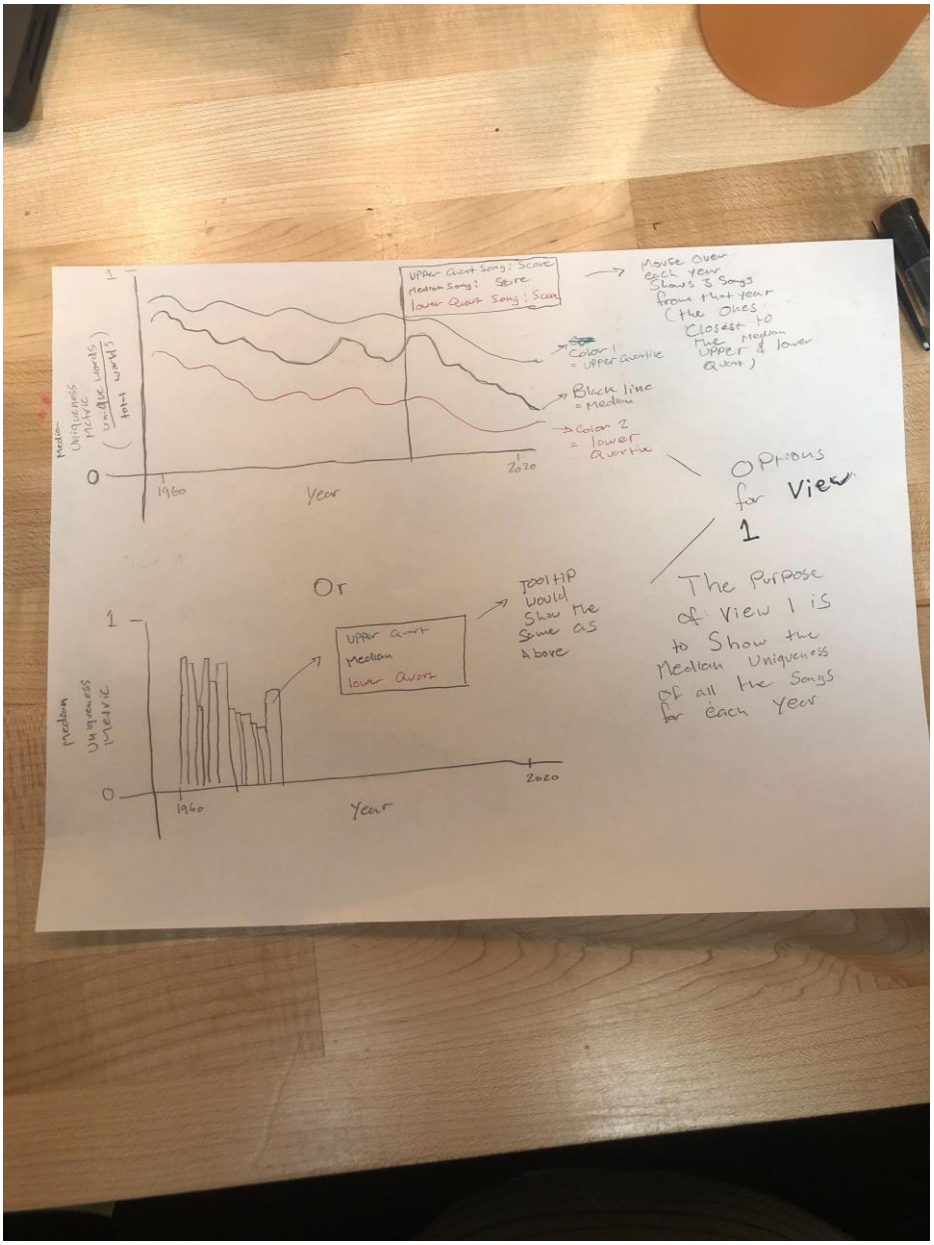
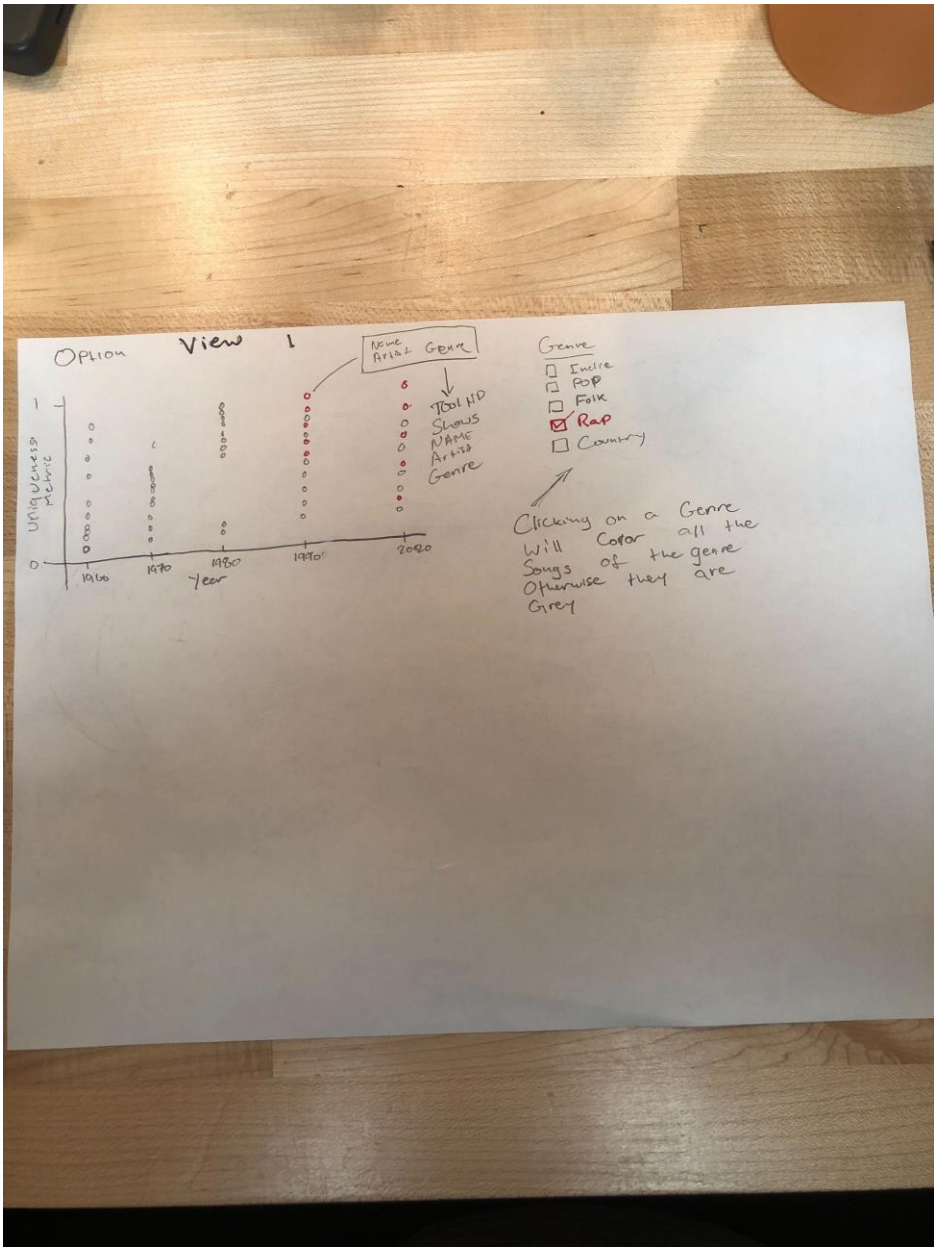

Upper Quart Song: Score
Median Song: Score
lower Quart Song: Score
Mouse over each year Shows 3 Songs from that year (the ones closest to the Median Upper & lower Quart)
Color 1 = Upper Quartile
Black line = Median
Color 2 = lower Quartile
Median Uniqueness Metric (Unique words / total words)
1
0
1960
Year
2020
Options for View 1
Or
TOOLTIP would Show the Same as above
Upper Quart
Median
lower Quart
The Purpose of View 1 is to Show the Median Uniqueness of all the Songs for each Year

Figure 3



Figures 4-6 attempt to visualize most frequent unique words in a genre/overall. We tried clustering based on genre/year and using size as a channel for frequency (see figures 5 and 6). We also tried using a simple bar chart (Figure 4), as well as create a chart with equal-sized bubbles where the position along the x-axis represented the frequency. The buttons represent interactivity to filter by genre.

Figure 4

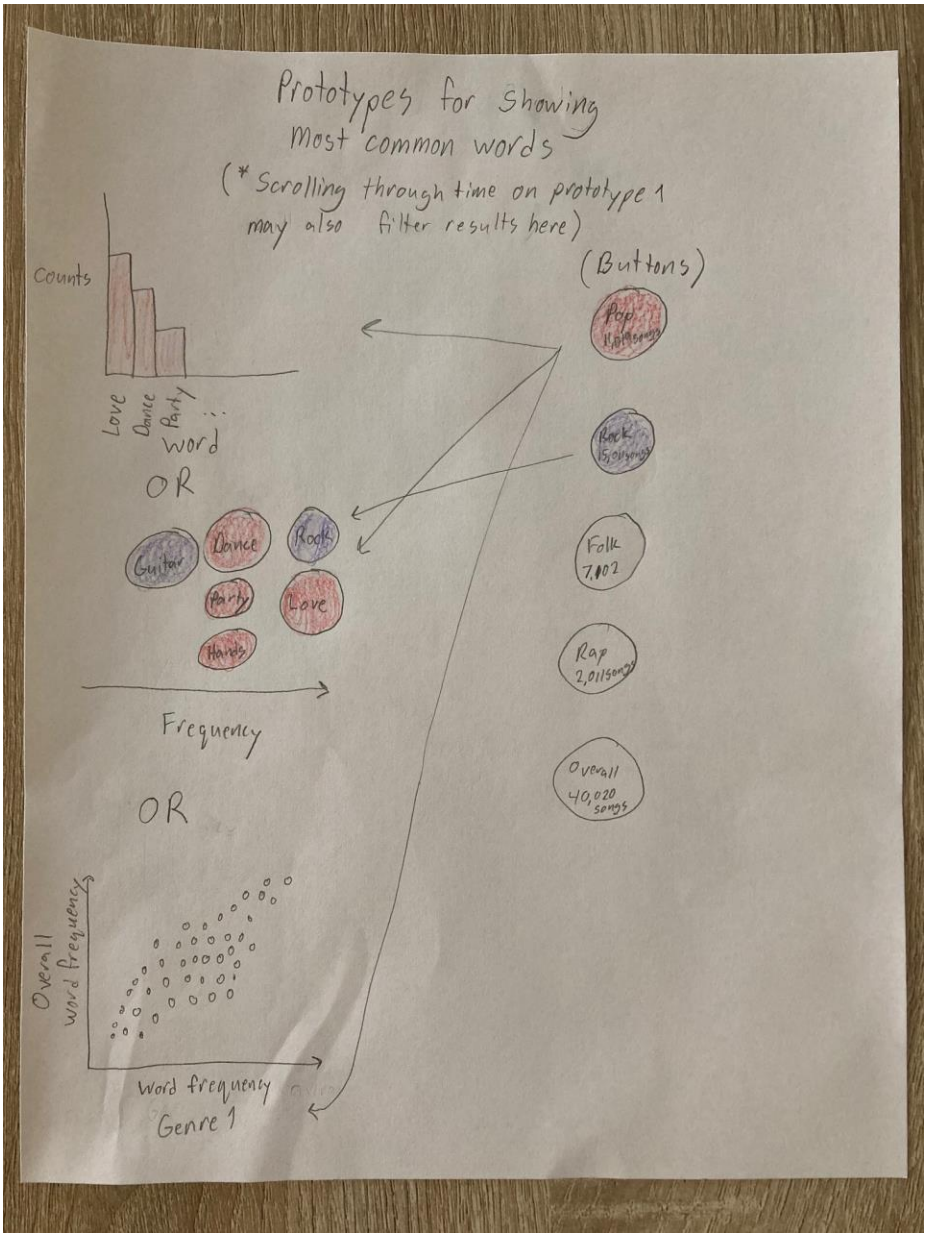

Prototypes for Showing Most common words
(* Scrolling through time on prototype 1 may also filter results here)
(Buttons)
Counts
Love
Dance
Party
...
Word
OR
Guitar
Dance
Rock
Party
Love
Hands
Frequency
OR
Pop
Rock
15,01 songs
Folk
7,902
Rap
2,011 songs
Overall
40,020 songs
Overall word frequency
Word frequency
Genre 1

Figure 5

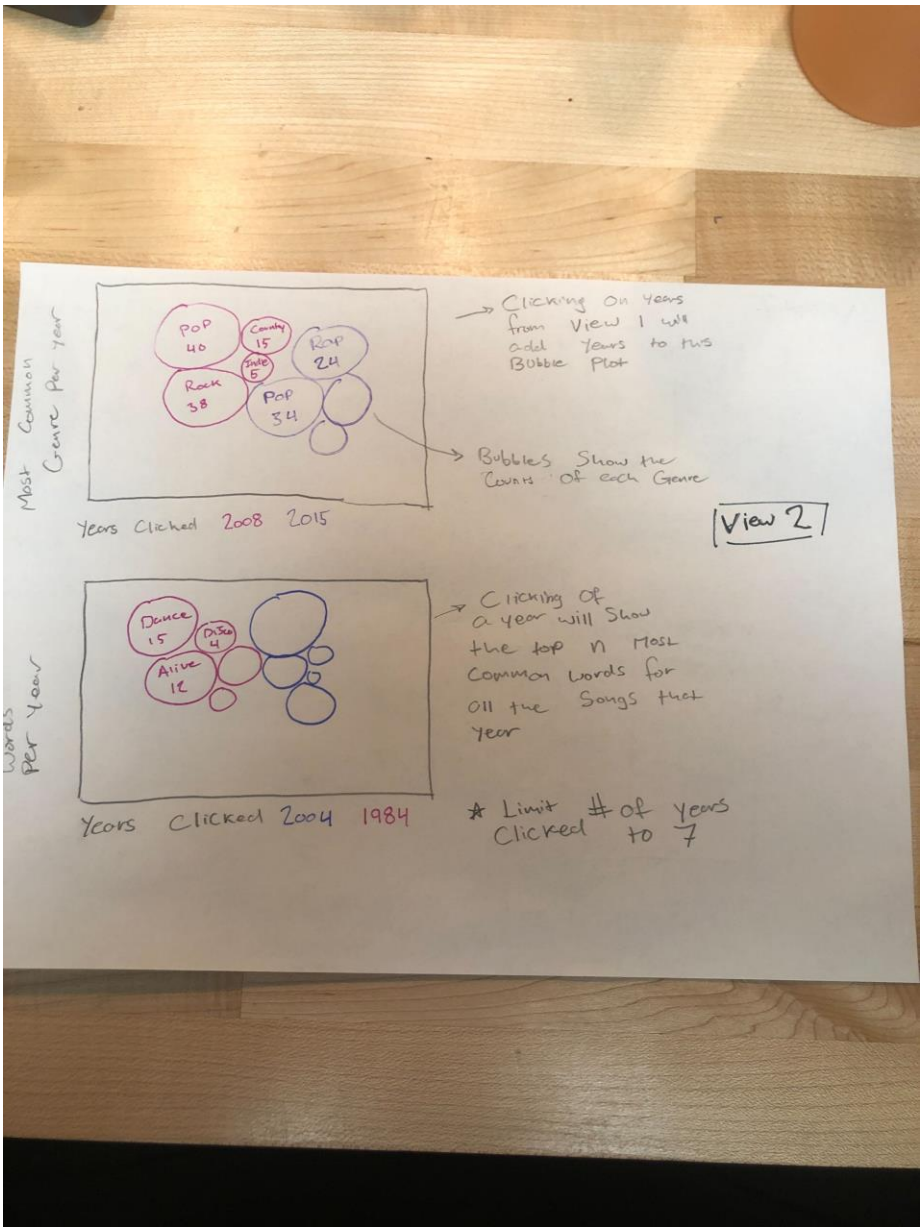

Most Common Genre Per Year
Pop 40
Country 15
Indie 5
Rock 38
Pop 24
Pop 34
Years Clicked 2008 2015
Clicking on years from View 1 will add years to this Bubble Plot
Bubbles Show the Counts of each Genre
View 2
Words Per Year
Dance 15
Disco 4
Alive 12
Years Clicked 2004 1984
Clicking of a year will show the top n most common words for all the songs that year
* Limit # of years Clicked to 7

Figure 6

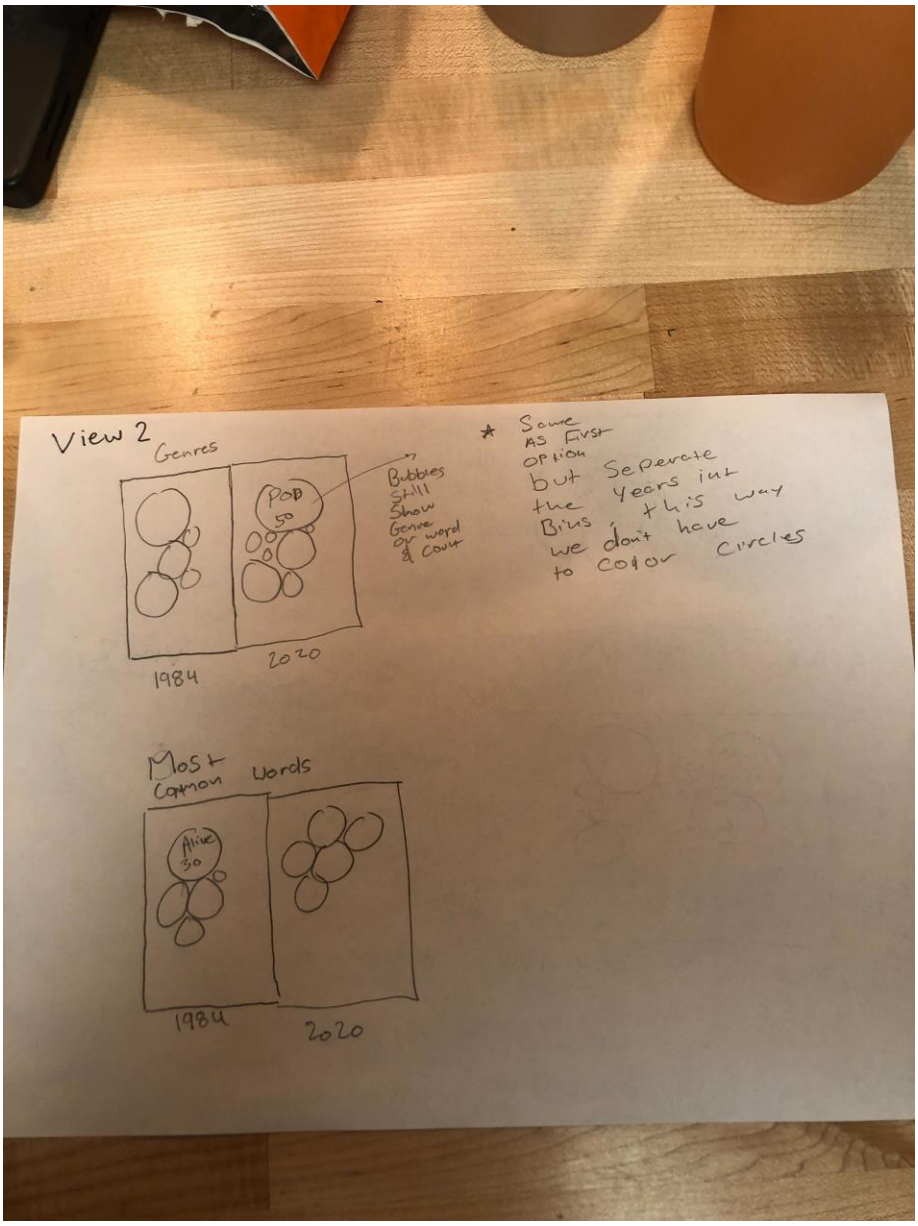


Figure 7 and 9 are attempts to showcase the change in word frequency for specific (predetermined) words over time. We tried using line charts and bar charts for accessibility and accuracy. We are also using color as a channel to differentiate between words. The checkboxes represent the ability to update the line chart based on the selected word.

Figure 7

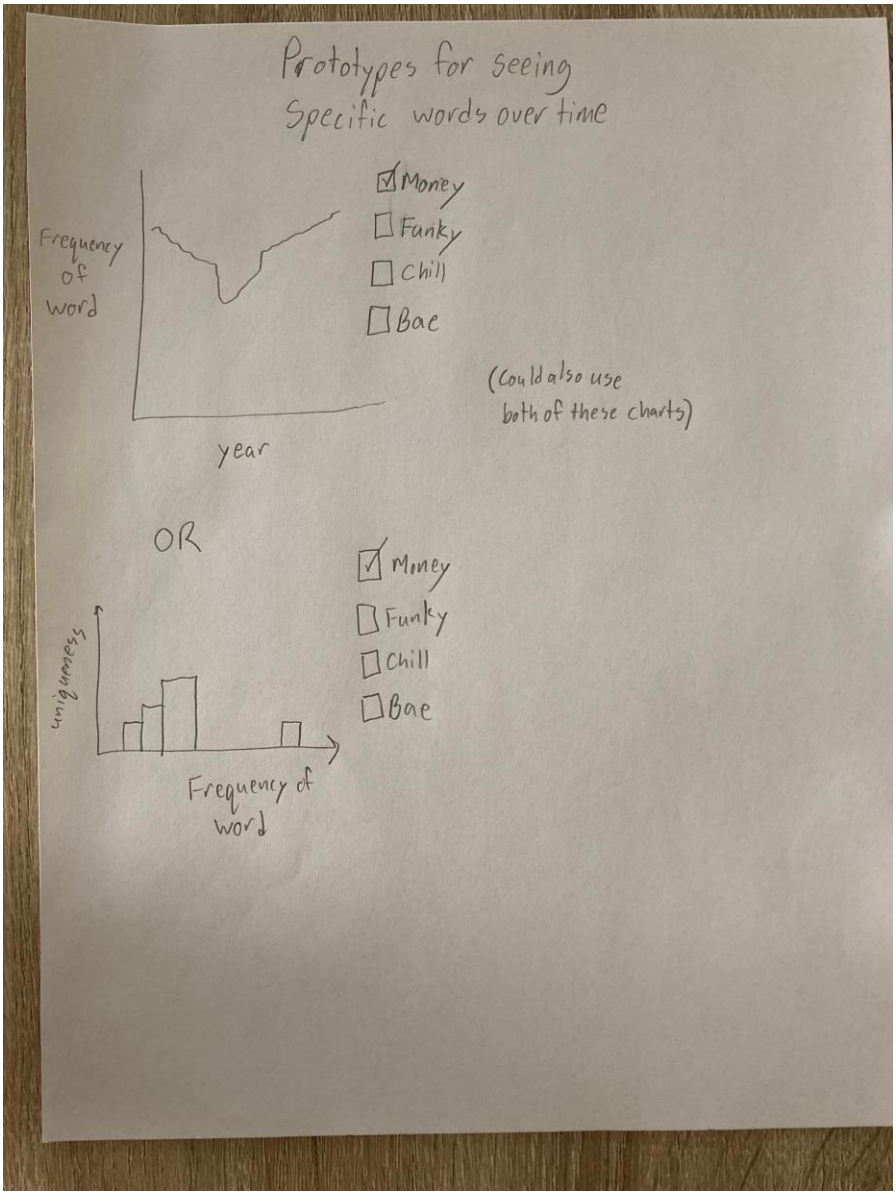

Prototypes for seeing
Specific words over time
Frequency of word
year
Money
Funky
Chill
Bae
(Could also use both of these charts)
OR
uniqueness
Frequency of word
Money
Funky
Chill
Bae

Figure 8

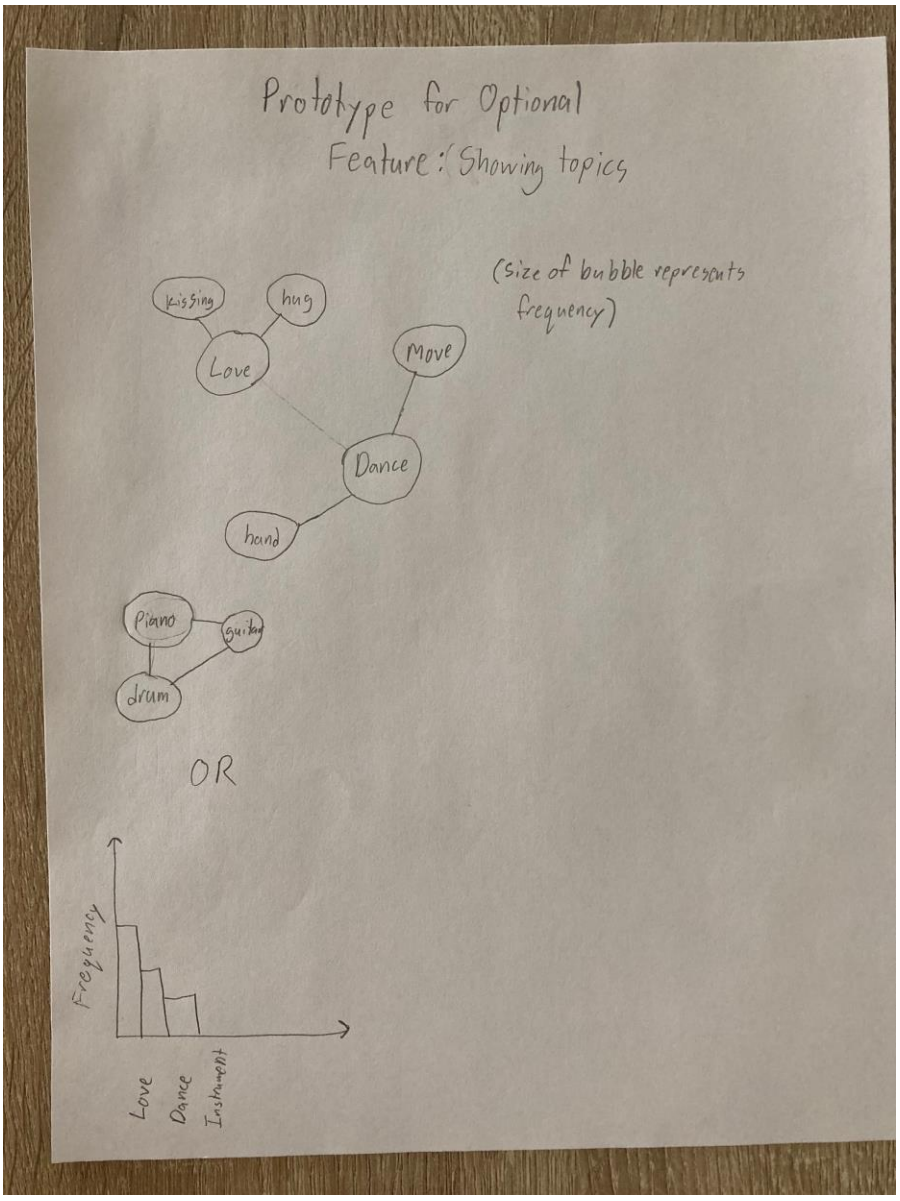
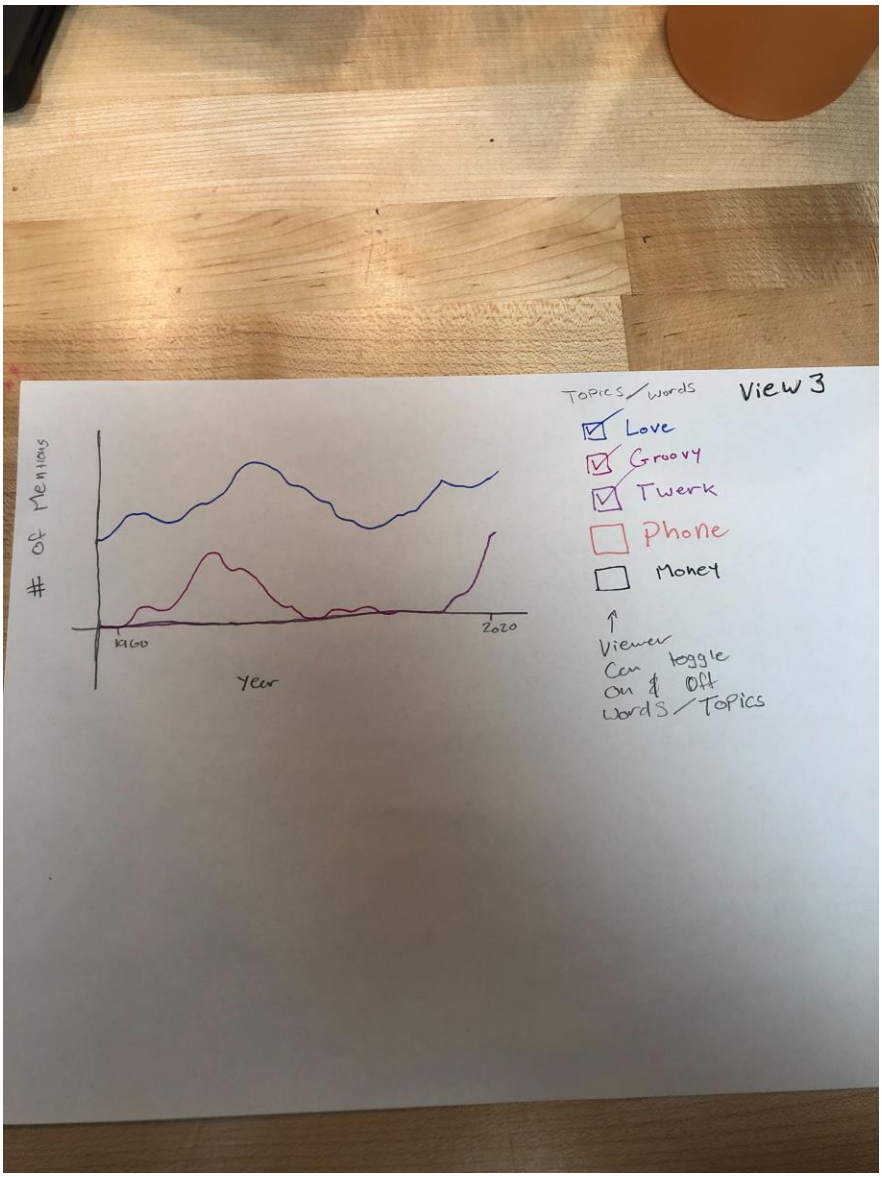

Prototype for Optional
Feature: Showing topics
(size of bubble represents frequency)
kissing
hug
Love
move
Dance
hand
Piano
guitar
drum
OR
Frequency
Love
Dance
Instrument

Figure 9



Main View:

For our master draft, we integrated and changed elements from our various prototypes. For our first view, we plan to create a line graph which shows our derived metric of uniqueness (unique word/total words) on the y-axis over time on the x-axis. The default will show a line corresponding to the entire integrated dataset. When selecting a button, the data will be filtered by genre, and this information will appear as an additional line on this graph with a different color, acting as an additional channel to differentiate between the genres/overall information.

In addition to the line graph, we will show a histogram, that shows the distribution of uniqueness per song (y-axis) and frequency (x-axis). This will also be filterable by genre.

This view was an integration of the 2nd and last prototype on figure 1 and 1st prototype on figure 2.

View 2:

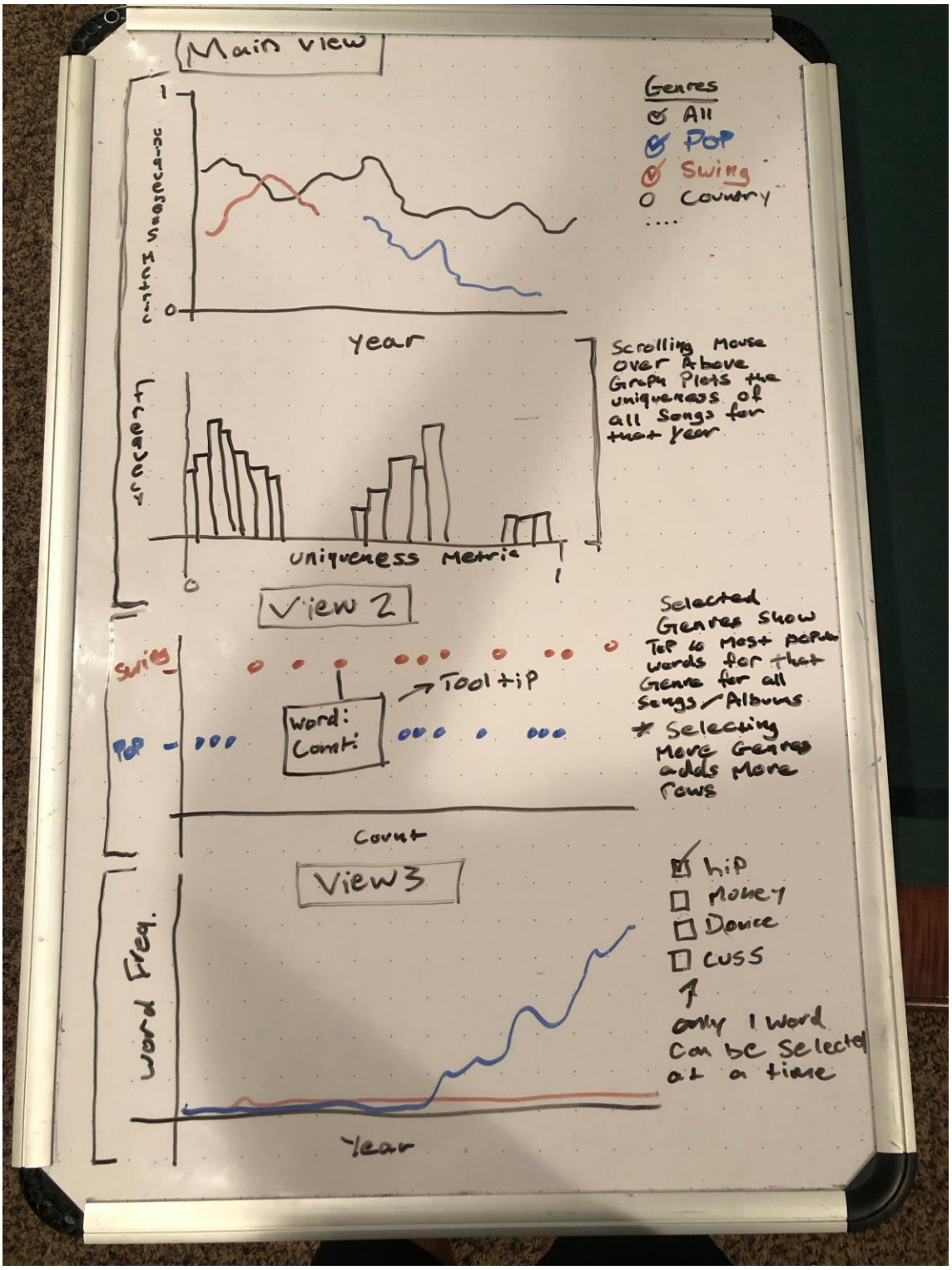
Our second view features a chart containing bubbles along an axis of word frequency (x-axis). The bubbles will contain a word, and the location along the x-axis will show the frequency of that word. The bubbles will ideally be organized by genre (and/or year), along with color-coding for the genre.

This view integrated figures 4-6.

View 3:

Our third view features a line chart with uniqueness (y-axis) and year (x-axis). This chart will show the change of specific (predetermined) word frequencies over time. This can also be filtered by genre (and possibly year).

This view comes from figures 7 and 9.



Must Have Features

List the features without which you would consider your project to be a failure.

Our project is minimally expected to showcase:

1) Visualizing the “unique word”/”total word” ratio throughout time. This will require the unique word counts and total word counts from the data processing, as well as the years from acclaimed music.
2) Visualizing the top used unique words by genre. This will require most popular unique word counts from data processing, as well as genre information.
3) Visualizing the counts of specific concepts or words like “Rock” or “Dance” over time. This will require us to count the instances of our key words for each song/album and organize this by genre.

Optional Features

List the features which you consider to be nice to have, but not critical.

We will optionally include:

1) Instead of exclusively visualizing top unique words, we may provide an option or create a graphic to display common themes in songs. This would require more substantial NLP to cluster unique words by similarity, and then would require gathering and displaying this data.
2) Options to filter by year. This will require organizing all the data by year in addition to genre – which will require a much larger and more structurally complex dataset. With this, we could extract the lyrics for the most repetitive/unique song for a given year if the user clicks on a year.

Project Schedule

Make sure that you plan your work so that you can avoid a big rush right before the final project deadline, and delegate different modules and responsibilities among your team members. Write this in terms of weekly deadlines.

Nov. 2nd

-Data is gathered from API

-Data is stored appropriately (json format)

Nov. 10th

– HTML page is created and structured

--Layout and svg’s created

--Data can be loaded/bound

--Hopefully axes and/or basic line graph created

Nov. 22nd

-First view created, including interactive elements

-Third view created, including interactive elements

Nov. 30th

-Second view created, including interactive elements

Dec. 2nd

-Project is finalized

-Miscellaneous elements taken care of, such as providing information when hovering, adjusting fonts/sizes, adjusting colors and sizing, etc.

Process Book

Data Gathering

The first task we had to tackle was gathering the data. As mentioned in our proposal, we planned to use the Genius API to gather song lyrics. The API calls straight forward. However, the lyrics that are returned need to be cleaned up a little bit. For example, all the lyrics come with a header that contains the title of the song that needs to be removed. After the lyrics are cleaned up, they are ready to be processed. We count the number of unique words and total words. We then used the NLTK package to remove the stop words from the lyrics before finding the most common words for each song.

After processing the lyrics, we still had one challenge in gathering the raw data. The Genius API has no way of obtaining the genre of each song. To solve this, we used the Spotify API to get the genre of the current artist. However, we found that the Spotify API returns the artist's genre in the following format.

['adult standards', 'brill building pop', 'easy listening', 'lounge']

This causes a few issues. What do we do with multiple genres per artist? What should we do about obscure genres that only show up a few times in the data? To solve these issues, we came up with the following solution. Using this website, https://www.chosic.com/list-of-music-genres/ we came up with 15 parent genres.

PARENT_GENRES = ["pop", 'rock', 'hip hop', 'latin', 'edm', 'r&b',

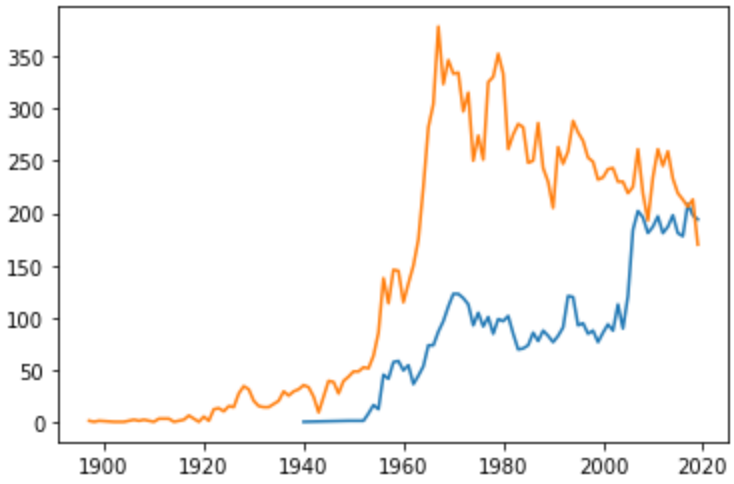
'country', 'folk', 'classical', 'metal', 'jazz', 'easy listening', 'new age', 'blues', 'world']

Each of these parent genres has a long list of child genres. We made a dictionary that mapped every child genre to its parent genre. Then when we call the Spotify API to get the list of genres. We loop through the output and keep track of how many returned genres map to each parent genre and we return the parent genre with the highest count.

For example, if an artist had 3 genres map to rock, and 4 genres map to pop, that artist would be classified as pop.

After all this we set up a pipeline to automatically gather all the songs. So far, we have gathered 400 songs, and we plan to get all 10,000.

As of right now, we are still deciding if we want to parse the most popular songs or the most popular albums. The songs CSV contains 10,000 songs, the albums CSV contains 3,000 albums, however, we estimate that each album will have about 5-10 songs, giving us a total of 15-30,000 songs to work with. The albums represent fewer years (line shown below in blue) than songs (orange). However, those years would likely contain more songs than the song CSV.



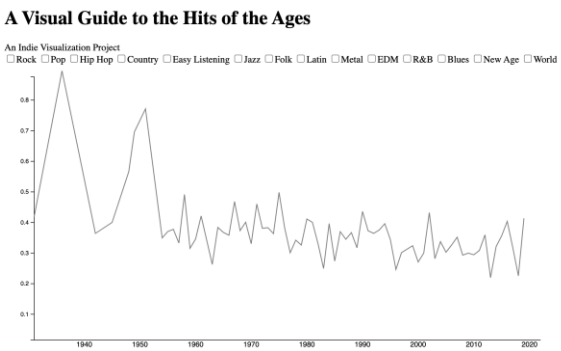
After writing a CSV of all the information that we wanted to extract, we used R to manipulate and filter the raw data. For example, some of the songs returned a raw word count of 10,000. I think in these cases, the Genius API couldn't find the lyrics for the song, so it returned an article about the song. These songs are filtered from our data. After cleaning the data, we used an R package to write the data frames into JSON files.

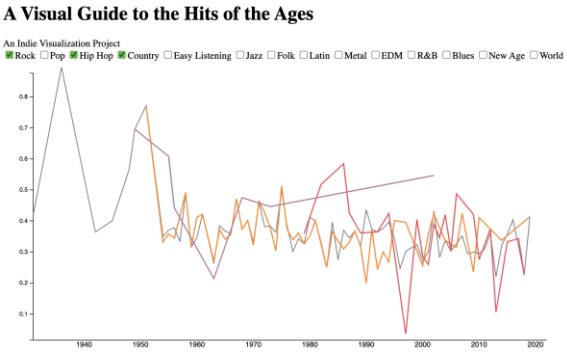
Creating View 1

View 1 will show the uniqueness metric over time for all the songs, and the selected genres.

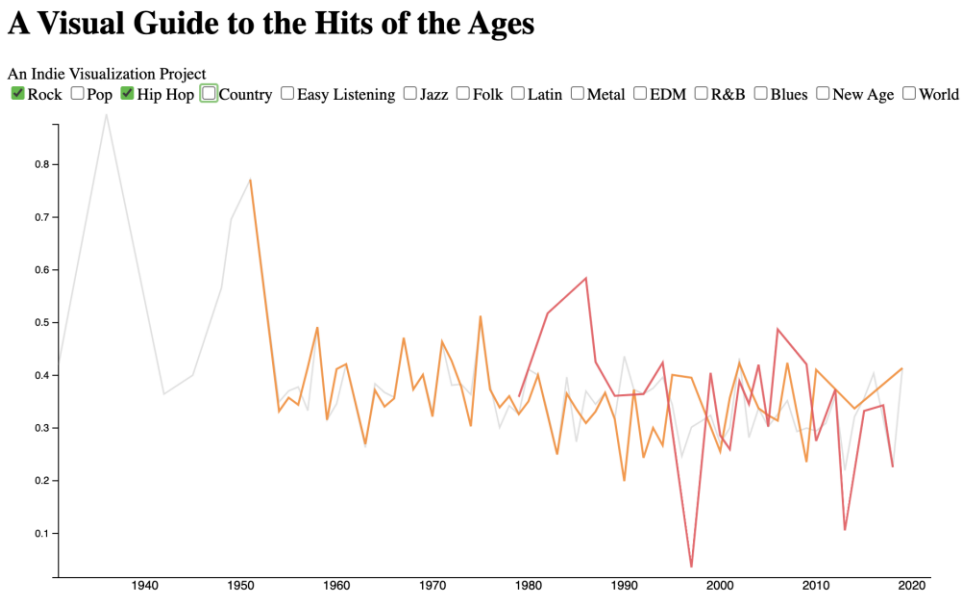
This is the first attempt at making View 1

The grey line represents the median uniqueness metric (y –axis) over all the years (x-axis). The user can toggle on an off different genre as shown on the right.





However, these feel a little cluttered to look at, so we lowered the opacity of the main line



We still need to add legends and axis titles to View 1.

Creating View 3

View 3 will show the average count of specific words per song – based on whatever word is selected.

The first attempt looks visually similar to View 1 (purposefully), though the data must account for the selected genres and for changing word choice. For this, the lines will be joined whenever there is a change in word and/or genre choice. The y-axis also scales to adjust for different magnitudes of the different words. Currently, the y-axis defaults to scaling the overall averages (the gray line), which does not always work with the specific genre selections (see end of second line graph).