Multimodal Sentiment Analysis of Billboard Hot 100's Lyrics and Audio Features

Project Report

Jeff Nguyen - 919619812
JT Herren - 919721771
Department of Statistics, University of California, Davis

April 7, 2024

Contents

Abstract

This project presents a multimodal sentiment analysis framework applied to the Billboard Hot 100 song corpus, integrating lyrical content and audio features to explore possible insightful correlations between differing musical components in contemporary music. By employing advanced natural language processing techniques, the research quantifies sentiment polarity in lyrics and correlates it with various acoustic properties obtained using the Spotify Web API. The findings reveal complex relationships between lyrical sentiment, musical composition, and the lack thereof for certain features, offering insights into the relationship between both.

1 Motivation/Introduction

The interest in this dataset stems from its comprehensive coverage of the diverse features of musical compositions. Generally these elements are not immediately apparent, but they carry great significance and allow for interpretation beyond what would be found by the average listener. One of these features stemming from musicality and lyrics is polarity, which refers to the inherent emotion or tone expressed within textual representations of lyrical music. Comparing this more subjective human interpretation to more inherent musical characteristics such as tempo, loudness, energy, and danceability allows for a deeper understanding of how these various elements shape the musicality of songs and how they intertwine with each other in the process of musical composition, which leads to greater interpretations of the significance music holds.

2 Methodologies and Challenges

This section dives into the specifics of the challenges we encountered during our working process. This project explored various methodologies including API usage, web-scraping, sentiment analysis, etc etc... The specific challenges attributed to a particular aspect of the project can be located under its respective subsection.

2.1 Data Collection and Processing

Data obtained from multiple sources were merged and processed, as detailed below.

2.1.1 Billboard API

The songs chosen for analysis in this project came from the Billboard Hot 100. Song data was retrieved using the Billboard API, which has a uniform format, being the title of the song and then the artists separated by the word 'by'. Using this consistency, the information was split into two parts: song and artist(s). Additionally, it was discovered that the featured artists were not necessary in the construction of the URLs for web-scraping purposes, and thus were subsequently removed.

2.1.2 Retrieving Lyrics from AZLyrics

Initially, AZLyrics was the online lyrical database chosen to scrape the lyrics for the songs listed in the Billboard Hot 100. The format of the HTML of AZLyrics was simple, making it relatively straightforward to retrieve the lyrics of a given song, or its English translation. Songs from the Billboard Hot 100 were formatted into the proper URL format for AZLyrics for scraping. Once the testing process of generating URLs and scraping lyrics for a single song was completed, an attempt was made to apply the logic to all 100 songs. After applying the function to a sample of the song data, results could be produced.

2.1.3 Retrieving Lyrics from Genius

Due to unforeseen circumstances, access to AZlyrics was lost. Genius was the next option considered to scrape lyrics from. A similar methodology to AZLyrics was employed for Genius, with the need for reformatting the list of 100 songs and their artists into proper URL format for Genius. Afterwards, those URLs were used to scrape for lyrics, notably by examining the HTML structure of Genius pages more rigorously than that of AZLyrics.

2.1.4 Selenium as a Scraping Alternative

After initial attempts to scrape lyrics from Genius were unsuccessful, Selenium was implemented as a possible alternative. In theory, this would be more streamlined in comparison to rigorous scraping, hopefully circumventing possible sources of error.

2.1.5 Improved Rate Limiting in Lyric Scraping

Taking the access denial of AZLyrics as a lesson, the rate limiter for scraping Genius was set to a request every 20 seconds. As a result, scraping has been successful thus far with no sign of access restrictions.

2.1.6 Cleaning and Formatting Genius Lyrics

Once lyrics were scraped from Genius for every song, the lyrics had to be cleaned and reformatted before a sentiment score could be calculated for a particular song's lyrics. This is because Genius formats their lyrics with the inclusion of headers specifying different sections of lyrical content, including but not limited to: Intro, Pre-Chorus, Chorus, Post-Chorus, Outro, and Verses. The removal of these extra headers was vital so that they are not considered as part of the lyrical content once it was time to undergo sentiment analysis.

2.1.7 Obtaining Audio Features of Songs using Spotify's Web API

To retrieve various audio features for all 100 songs in the Billboard Hot 100, spotipy was utilized, a lightweight Python library for the Spotify Web API. By navigating to the Spotify for Developers page, creating an account allowed for an easy access token for our purposes. After reading the documentation and writing the audio features fetching function by querying song titles and artist names from our dictionary of all 100 songs, information was able to be gathered. These audio features of individual songs include:

- danceability
- energy
- key

- loudness
- mode
- speechiness
- acousticness
- instrumentalness
- liveness
- valence
- tempo

2.1.8 Performing Sentiment Analysis

This project's approach to sentiment analysis is a simplified version. More robust sentiment analysis methods involving machine-learning (ML), as well as utilizing pre-trained deep-learning models were initially considered, however, ultimately it was decided to use TextBlob, a lexicon-based sentiment analyser used by leveraging parts of NLTK (Natural Language Toolkit) for its implementation. TextBlob is simple to use and the information it provides is easily digestible. After briefly reading its documentation, a simple function was written to retrieve sentiment scores for the textual lyrics of a particular song. These sentiment scores include polarity and subjectivity, the former being scored from -1.0 to 1.0, and the latter 0.0 to 1.0. In short, the further the polarity score of a set of textual lyrics is below zero, the more negative the sentiment of the text, and vice versa. For the subjectivity score, a score closer to 0.0 indicates objective and fact-based information whilst a score closer to 1.0 indicates subjective and heavily opinionated information. Such scores were found for every song in the list and their overall sentiment was decided relative to their polarity scores around zero.

TextBlob's sentiment analysis relies on a predefined lexicon where each word is associated with its respective polarity and sentiment score. Once implemented, TextBlob divides a piece of text into its constituent words and looks up sentiment scores for each from the lexicon. It then calculates an aggregate score based on the words' individual scores to determine the text's overall sentiment numerically.

2.1.9 Formatting and Merging Gathered Data

At this point, the data existed in separate formats, with the Spotify data being a list of dictionaries each containing the audio features of interest of a song, while the other format consisted of a dictionary with the song title as a key and the lyrics as a value. To merge these data formats, it was necessary to identify matching keys between all of the dictionaries. Using the shared key of 'song title' it was possible to create a singular data format. The final dictionary format was a dictionary with the song title as the key with the value being a dictionary that contained the values of interest for each song. These features included Artist, Overall Sentiment, Polarity, Subjectivity, Valence, Energy, Loudness, Tempo, and Danceability. With this combined format the dictionary could be read into a Pandas dataframe.

2.1.10 Visualizing the Data

With a completely formatted dataframe it was necessary to visualize the data to better interpret its contents. A heatplot was chosen as the first model because it gives a simple and direct representation of the correlation between variables based on their values. This process required isolating the specific columns of interest, which encompassed all the quantitative variables. The created plot gave a value from -1 to 1 that determines the correlation each variable has with the others. The next chosen visualization was a scatterplot because its ability to visually highlight trends in correlation more dynamically allows for more potential to explore relationships in the data correlation instead of narrowing it down to a single number. (If I can figure out the interactive scatterplot) Finally, an interactive scatterplot was included as well in the instance that one would want to investigate trends between lyrical sentiment and audio features for a particular song in the Billboard Hot 100. An interactive plot was also generated, where one can hover over individual data points to see details for a specific song, including song title and artist(s). An additional feature added to the interactive scatterplot was the ability to interchangeably set the variables for each axis to explore relationships between different variables easily.

2.2 Challenges and Solutions

Many obstacles arose during the implementation process. This section details the solutions, improvisations, and problem-solving necessary for the successful completion of the project.

2.2.1 Working with AZLyrics and Loss of Access

There was some difficulty getting the song information retrieved from the Billboard API into the proper AZLyrics URL format for scraping purposes. Once it was understood that the URL for a given song in AZLyrics is structured as the main artist and song title with no spaces or special characters, all 100 songs could be formatted into their proper URLs. A few setbacks arose due to inconsistencies in the song data as given by Billboard, but after some string manipulation, success was found.

Unfortunately, when attempting the scraping process on all 100 songs, the requests to AZLyrics were not adequately rate limited. As a result of this, access to the lyrics database was lost. This knowledge was taken into consideration for the next attempt at scraping lyrics. As the Billboard Hot 100 contains many recent songs, whichever new database used needed to contain lyrics for recently released music as well. This was a challenge as many lyrics databases contained only popular songs or old songs. However, new suitable option was eventually found, pivoting the scraping process to Genius.

2.2.2 Reformatting URLs and Scraping Content from Genius

Moving to a new lyrical database meant having to alter the previously created logic for extracting song data and creating URLs for AZLyrics. The format of Genius URLs is noticeably more complex in structure. All non-featured artists are included in the URL, where any special characters and whitespace are substituted either with their respective letter equivalents or with dashes. The function that formats song data to ensure that all songs could be properly incorporated into the new URL standard had to be slightly altered. Additionally, the URL creation function had to be changed to accommodate the new rules. This process included replacing special characters such as $ and '@' with their corresponding counterparts 'S' and 'at' respectively. After these alterations, Genius lyric links to all the songs in the Billboard data were able to be successfully created.

The main issue encountered when it came to the actual scraping for lyrics from Genius was the page structure. An initial attempt at scraping the lyrics returned only a portion of the lyrical content instead of the entire song. At first, the origin of this problem could not be identified, and so the implementation and usage of Selenium was attempted in place of rigorous scraping, however, the challenges encountered with Selenium are detailed in the following section. After examining the page structure closely through inspecting element, it was revealed that the lyrics were spread out across multiple containers, and so the function logic had to be adjusted accordingly to retrieve the lyrics of each song in its entirety.

2.2.3 Crashing Issues with Selenium

This was the first attempt at using Selenium in general, and so many issues were encountered in the process of its implementation. Due to one way or another, every attempt at using Selenium was unsuccessful. Being a quite wide scope of obstacles, the details are concisely summarized in the following paragraph.

At first, the wrong version of the chromedriver was downloaded, being incompatible with the current version of Google Chrome used. Afterward, installing the package, importing it, and attempting to run the module ended up instantly crashing chromedriver. Additional assistance packages such as webmanager were proven ultimately unhelpful. Rearranging file paths and specifying the paths for Google Chrome and chromedriver in system variables also resulted in a failed implementation. With these setbacks sequentially appearing, a decision was made to pivot back to refining the rigorous web-scraper for Genius. At this point, it is important to note that the frustration that came with the many failed attempts to implement Selenium almost swayed the lyrical fetching in the direction of directly using the Genius API to obtain them, however, there was strong motivation to demonstrate knowledge of obtaining information in a variety of ways.

2.2.4 The Caveat of Rate Limiting

While the rate limiter being set to 20 seconds helped the access issue, it raised a separate concern. The rate limiter being so lengthy resulted in the retrieving of lyrics also taking a considerable amount of time, approximately 35 minutes of runtime for a complete test run. This caused some time delays due to the collaborative notebook environment used, which would occasionally reset variables if left alone for too long, impacting the timeliness of the data formatting for example.

2.2.5 Brackets, Parentheses, and Newlines in Genius Lyrics

When it came time to clean and reformat the lyrics scraped from Genius, there were a myriad of issues that arose in the process. The first and easiest to resolve were the headers separating different song lyrical sections. These headers were typically put inside brackets, and thus regex expressions were utilized to remove any brackets and the text contained within them.

The next and largest challenge in the lyric cleaning and reformatting process was a result of how the background vocalizations in certain songs were scraped from Genius. Background vocalizations are typically formatted as italicized text contained within parentheses placed at the end of a lyric line. When scraped, however, the background vocalizations would be split among numerous new lines, with the left and right parentheses and vocalization text all being printed on separate lines.

A preliminary consideration was if it would be a later issue to keep the background vocalization on its new line after collapsing the parenthetical expression. At this point, a decision had to be made whether or not to include background vocalizations in the textual lyrics in the first place, as having the background vocalizations on a separate line could interfere with the textual sentiment analyzer, considering it as a separate entity as the previous line it was supposed to be attached to, as the context of these vocalizations played into how they are perceived emotionally. Ultimately, they were decided to be kept, however, with the condition that the parenthetical expression had to be removed from the new line and then appended to the previous line without affecting any other subsequent lyrics that came after. This led to complex modifications of the logic inside the lyrics cleaner function as well as the regex expressions necessary alongside. Upon initial testing, parenthetical background vocalizations had been successfully removed and appended to the previous line as intended, however, it would also impact subsequent lyrics that came after, combining lyrics unnecessarily. With further fine-tuning of the regex expressions, lyrics were able to be cleaned and reformatted as intended.

2.2.6 Discrepancies in Songs and their Complications with Spotipy

During the testing stage of creating the audio feature fetching function, there were several issues when it came to querying song titles and artists' names to retrieve information. There were a few discrepancies in naming conventions, especially when it came to remixes of songs, multiple versions of the same song, and cases where special characters were present in artist names. The function logic had to be rewritten and adapted accordingly, however, thankfully these obstacles in testing did not prove to be a wide issue when it came time to iterate through the entire list of 100 songs.

After retrieving audio features for every song, a main discussion point where challenges were faced was which of the audio features would be relevant for this particular sentiment analysis, i.e. which features to consider and exclude moving forward. Given that the primary goal was to discover potential correlations between audio features and lyrical sentiment, it was decided that the features that would be used for analysis would be Danceability, Tempo, Valence, Loudness, and Energy, as these were elements that could be more intertwined with the negativity or positivity of a song's lyrics than say, key. It is important to note that valence is Spotify's own measure of a song's overall positivity, scored from 0.0 to 1.0, with increasing scores indicating higher positivity levels.

2.2.7 The Simplicity of TextBlob

As previously mentioned, TextBlob, a lexicon-based sentiment analyzer was utilized in this project instead of its more complex alternatives, primarily due to a lack of experience and knowledge, thus simplicity and concrete understanding were prioritized. Due to this, there are several cautions to consider when interpreting the results of the sentiment analysis performed. While TextBlob is simple and fast whilst not needing training data, it is also in its simplicity that analyses obtained from the data it produces may not be as purely reliable as more complex alternative techniques. Unlike utilizing machine-learning or a pre-trained deep-learning model, TextBlob may not accurately capture sentiment in textual data where context drastically changes meanings in words. Additionally, the fixed nature of the lexicon it relies on means that it may not account for new slang, expressions, or the natural evolution of semantics. For projects requiring more nuanced techniques for texts with more complex linguistic structures, more advanced methods should be considered, however, the lack of training data and computational resources available for this project demonstrates the reliability and need for a straightforward method that is easier to use.

2.2.8 Key Matching Within Data Merging

During the formatting process, there were some setbacks caused by the shared notebook being used. While attempting to properly format the data the notebook would sometimes reset the variables. This, combined with the increased rate limit for obtaining lyrics, led to delays while testing. Aside from that there was some difficulty matching the keys between the dictionaries because the lyric dictionary contained the song titles as shown in Billboard while the audio features dictionaries contained the song titles as stored in Spotify. After some alterations, it was possible to match the keys of the dictionaries.

2.2.9 Sharing Visualizations

An initial challenge was figuring out how to share the interactive scatterplot with readers due to the limitations of a PDF format. It was initially decided to utilize Chart Studio by Plotly to host the scatterplot under their servers where a link could be provided to view it, with the caveat that only the hoverable data points were possible. For interchangeable axis variables, the source code will be provided in the code appendix for those wishing to explore visual correlations in more depth within the program itself, however, while Plotly was able to generate an interactive plot in the Jupyter environment, attempts to host a web app of the plot for others to use using Dash were unsuccessful, so Streamlit was later used as an alternative hosting service to resolve this problem.

3 Preliminary Interpretation of Data

Our preliminary analysis of the data showed that the overall sentiment of all the songs in the data where around 50/50 positive and negative, which would suggest that if a relationship was found amongst the different features it would be indicative of both sides of the sentiment spectrum. Additionally, it was apparent that some of the audio feature values were significantly larger than those of the sentiment analysis which may lead to lower correlation coefficients even if the two variables have strong relationships with each other.

3.1 Processed Data

Table 1: Subset of Song Dataframe

	Song	Artist	Overall Sentiment	Polarity	Subjectivity
0	Carnival	¥$	Positive	0.010955	0.568098
1	Lose Control	Teddy Swims	Positive	0.041739	0.410426
2	Lovin On Me	Jack Harlow	Negative	-0.041799	0.424699
3	Beautiful Things	Benson Boone	Positive	0.385714	0.686905
4	Texas Hold Em	Beyonce	Negative	-0.063019	0.460030
5	Greedy	Tate McRae	Negative	-0.029240	0.604678
6	I Remember Everything	Zach Bryan	Positive	0.076144	0.625817
7	Snooze	SZA	Positive	0.144133	0.568240
8	Agora Hills	Doja Cat	Negative	-0.032440	0.509539
9	Cruel Summer	Taylor Swift	Negative	-0.104993	0.559264

Table 2: Audio Features Subset

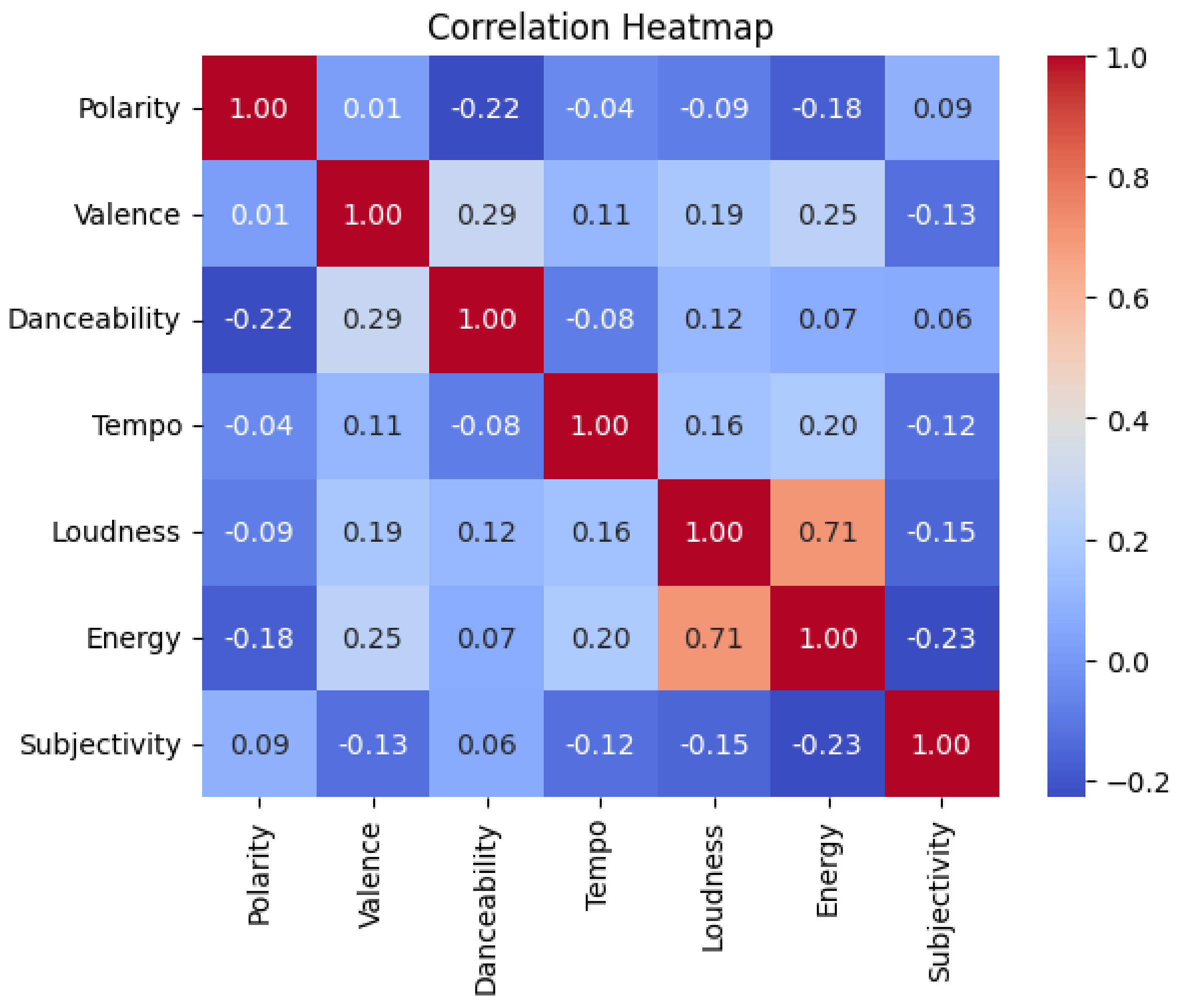
	Song	Valence	Energy	Loudness	Tempo	Danceability
0	Carnival	0.311000	0.811000	-5.746000	148.144000	0.594000
1	Lose Control	0.242000	0.604000	-4.409000	159.920000	0.561000
2	Lovin On Me	0.606000	0.558000	-4.911000	104.983000	0.943000
3	Beautiful Things	0.219000	0.471000	-5.692000	105.029000	0.472000
4	Texas Hold Em	0.353000	0.709000	-6.514000	110.024000	0.725000
5	Greedy	0.844000	0.733000	-3.180000	111.018000	0.750000
6	I Remember Everything	0.155000	0.453000	-7.746000	77.639000	0.429000
7	Snooze	0.392000	0.551000	-7.231000	143.008000	0.559000
8	Agora Hills	0.392000	0.674000	-6.128000	123.026000	0.750000
9	Cruel Summer	0.564000	0.702000	-5.707000	169.994000	0.552000

3.2 Dataframe Overview

As seen in the table subsets of the DataFrame, each row represents a specific song and contains the associated characteristics of interest. First of the categorical variables include the main artist of the song and the overall sentiment of the song, which is based on the polarity in relation to its distance from zero. Moving on to the quantitative variables, there are polarity and subjectivity, which result from the sentiment analysis of lyrics, and represent the emotion expressed by the song and its inherent bias respectively. The rest of the audio features retrieved from Spotify are based on the sound of each song except for valence which is Spotify's own way of categorizing the song's positivity.

3.3 Data Visualization

3.3.1 Heatmap



	Polarity	Valence	Danceability	Tempo	Loudness	Energy	Subjectivity
Polarity	1.00	0.01	-0.22	-0.04	-0.09	-0.18	0.09
Valence	0.01	1.00	0.29	0.11	0.19	0.25	-0.13
Danceability	-0.22	0.29	1.00	-0.08	0.12	0.07	0.06
Tempo	-0.04	0.11	-0.08	1.00	0.16	0.20	-0.12
Loudness	-0.09	0.19	0.12	0.16	1.00	0.71	-0.15
Energy	-0.18	0.25	0.07	0.20	0.71	1.00	-0.23
Subjectivity	0.09	-0.13	0.06	-0.12	-0.15	-0.23	1.00

Figure 1: Correlation of Sentiment Scores with Audio Features

Based on the heatmap, the values gathered through sentiment analysis did not appear to have a significant correlation with the audio features. The highest correlations for polarity and subjectivity were danceability and energy respectively, but both correlation values fell below 0.5 meaning that the correlation between the variables is weak. They are beyond the threshold of having no correlation, but not enough to make significant claims from. The highest correlation found from the heatplot was between the two audio features loudness and energy with a value of 0.71. This suggests a strong linear relationship between these two variables meaning as one increases it is likely the other would as well.

3.3.2 Example Scatterplots

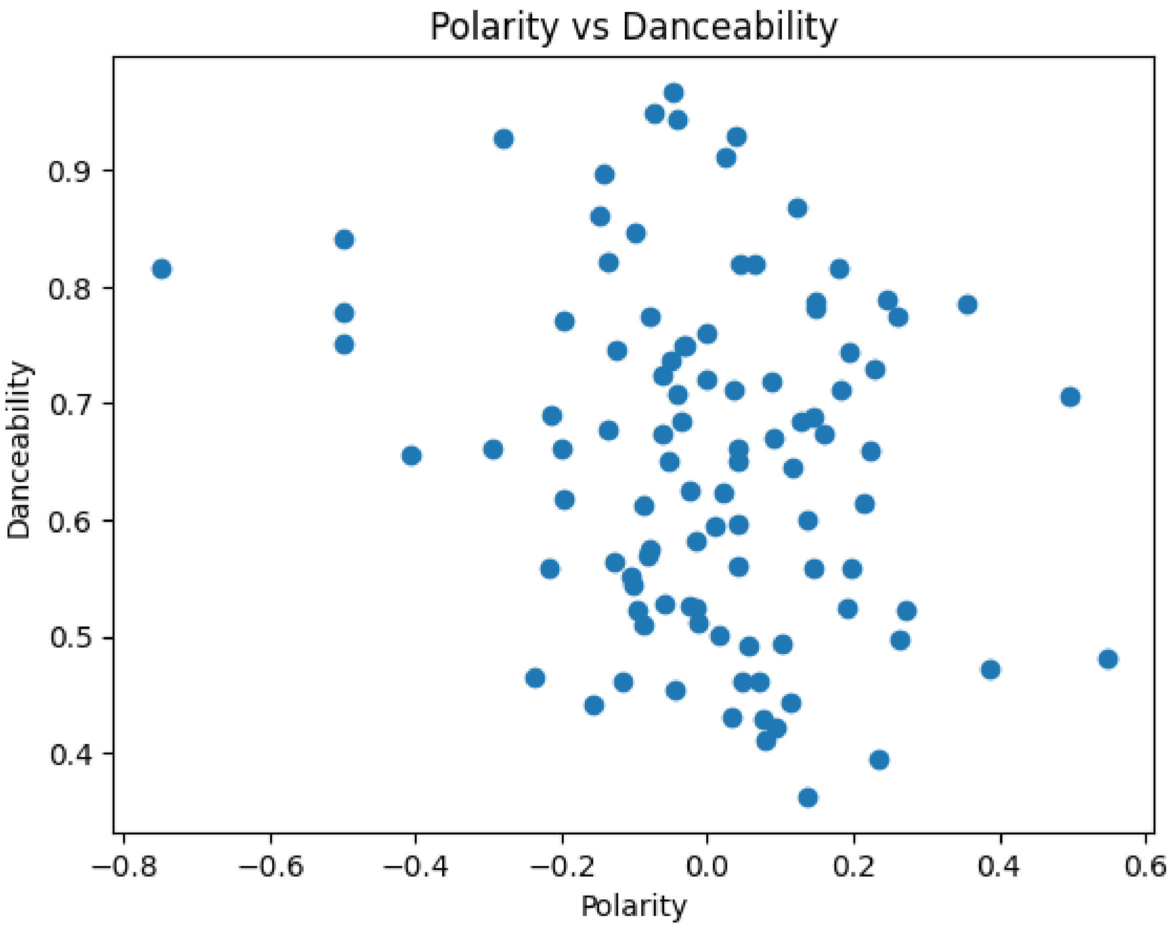


Figure 2: Correlation of Polarity and Danceability

The scatterplot of Polarity versus Danceability appears to have a weak positive correlation because the majority of the data points appear to loosely create a diagonal line. Due to the data being rather spread out it would suggest that it is unlikely that there is a strong linear relationship between these two variables. This would mean that these variables may have a non-linear relationship. However, when it comes to direct correlation the connection is weak.

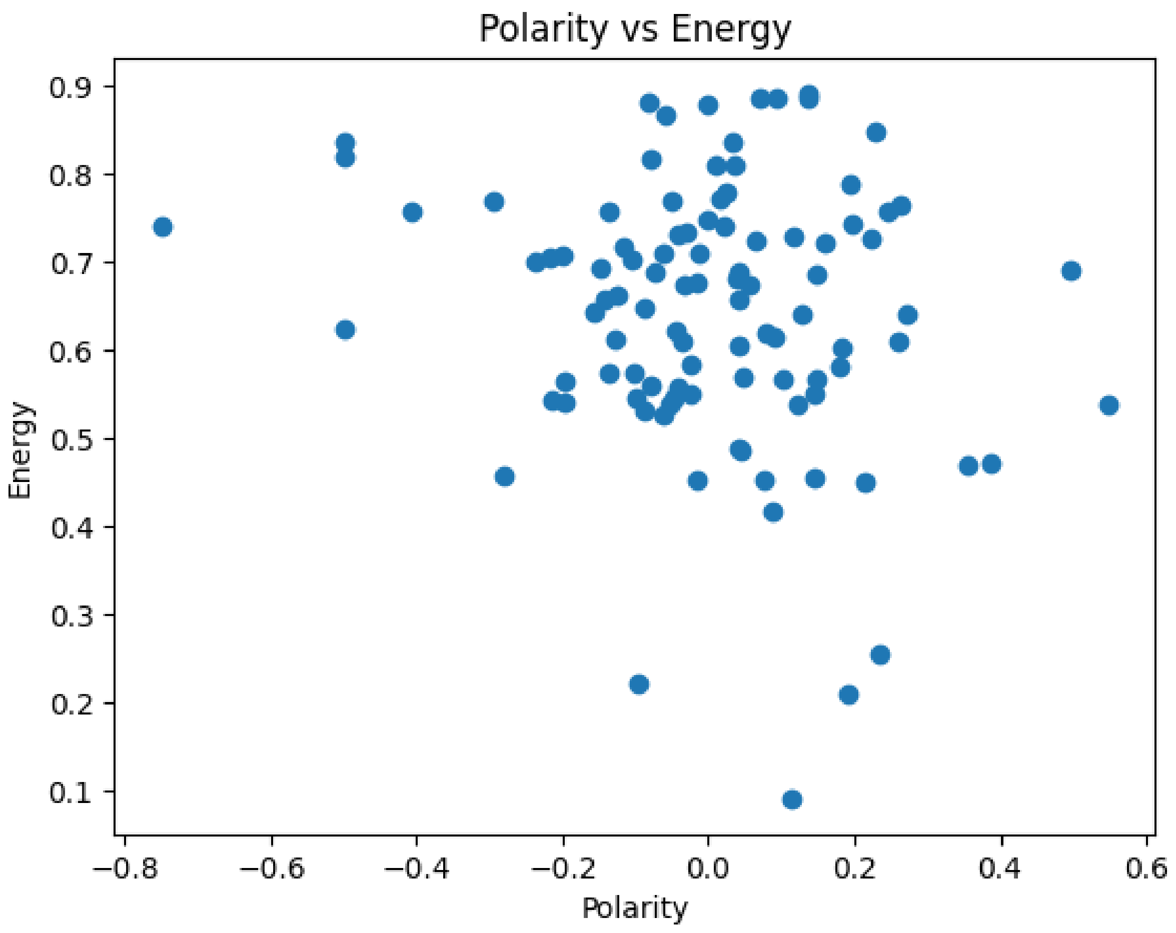


Figure 3: Correlation of Polarity and Energy

The spread of the scatterplot for Polarity versus energy appears to be similar to the previous plot. However, most of the data points in this plot are more clustered together making it appear to have a more clear linear pattern. Also, the outliers present are much more apparent than for the previous plot. This would suggest that if it would be worth investigating the effect of outliers. For the data as a whole, it would appear that these two variables have a weak linear relationship, but if outliers were taken into account in the overall analysis it is possible a stronger relationship could be found.

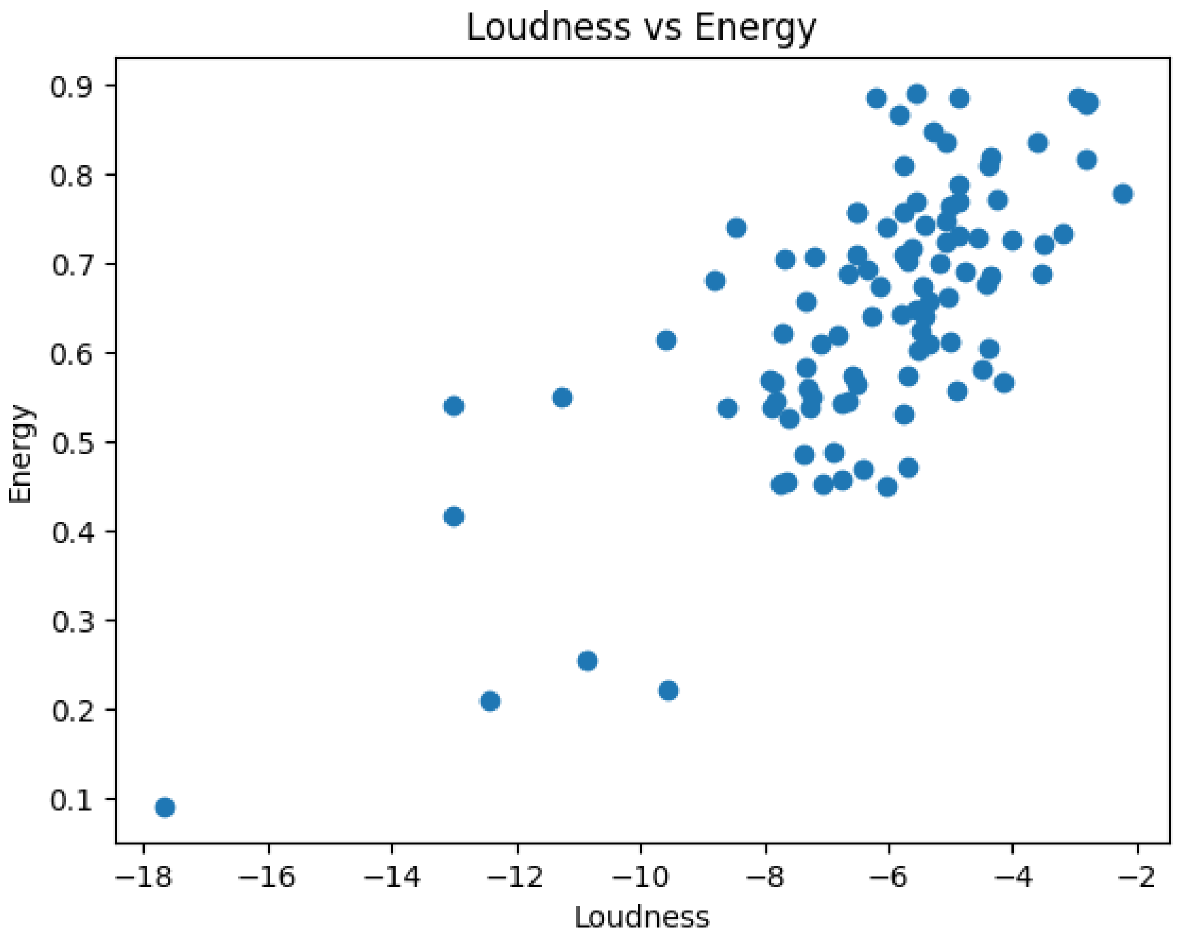


Figure 4: Correlation of Loudness and Energy

Although the main point of interest was exploring relationships between audio features and the sentiment polarity score specifically, a graph of Loudness verses Energy is provided as an example of a stronger correlation, as previously noted in the heatmap. The scatterplot for Loudness Versus Energy has a very clear linear pattern. This would suggest that Loudness and Energy have a very strong positive correlation meaning as one of them increases it is very likely that the other would as well, which intuitively is unsurprising.

3.3.3 Interactive Scatterplot

Due to the limitations of PDF formatting, the interactive visualizations are provided externally. There are quite a number of combinations of possible variables to plot together, so readers are invited to explore relationships at their own discretion. The plots of most significance were discussed in the previous section.

To view the interactive scatterplot with hoverable data points, click here.

To view the improved interactive scatterplot with interchangeable axis variables, click here.

4 Conclusion

In conclusion, our analysis revealed that there is no significant evidence to conclude definitively that values gathered through sentiment analysis have a strong correlation with different features of musicality. It was discovered that the highest correlated variables were still too weak to create a conclusive claim. Despite a failure to find a strong linear correlation, there were some takeaways. focusing on Polarity and Energy it would seem that there could be a direct correlation given some manipulation of the data. A possible way to test this could be through the use of different or larger amounts of data. Another point of interest is that polarity and tempo have very different scales and that if tempo were normalized it is possible that some correlation could be found between the two variables. It is also possible that some of these variables have non-linear relationships, but further analysis and alternate methods would be required to form a conclusive claim.

5 File Appendix

To view and/or download resources used in this project, including the source code, ipynb file, visualizations, or the dataset, click here.

To navigate to the GitHub repository for the interactive plot web-app, click here.

References

[1] *AZLyrics*. URL: https://www.azlyrics.com/.

[2] *Billboard*. Mar. 2024. URL: https://www.billboard.com/charts/hot-100/.

[3] *Genius*. URL: https://genius.com/.

[4] Allen "guoguo12" Guo. *Guoguo12/Billboard-charts: Python API for downloading Billboard charts*. URL: https://github.com/guoguo12/billboard-charts.

[5] Baiju Muthukadan. *Selenium with python¶*. URL: https://selenium-python.readthedocs.io/.

[6] *Plotly Documentation*. URL: https://plotly.com/python/.

[7] *Dash with Plotly*. URL: https://dash.plotly.com/.

[8] *TextBlob Documentation*. URL: https://textblob.readthedocs.io/en/dev/.

[9] *Spotipy Documentation*. URL: https://spotipy.readthedocs.io/en/2.22.1/.

[10] *Streamlit Documentation*. URL: https://docs.streamlit.io/.