

Project 2 Write-up
INFO 3510 Music as Information
Fall 2025

After finishing project 1, I had mixed feelings. On one hand, I had successfully analyzed sentiment noticing differences between living and deceased emo rap artists. On the other hand, I knew I'd relied too heavily on Claude AI to help me with refactoring and uncomplicating code I had written, where some parts were too efficient and advanced for my own understanding. I could run it, I could tweak it, but if someone had asked me to explain every line in detail, I would have struggled. That realization was uncomfortable but important.

For Project 2, I made a very conscious decision that I wanted to build something interactive that extended my first project, but this time... I needed to truly understand every piece of code I wrote. Even if that meant building something simpler. Even if that meant saying no to fancy features I didn't fully grasp. My main research question here was "can I design an interactive tool that lets people explore sentiment in song lyrics in real time, while also forcing myself to fully understand every part of the code?"

I decided to build a Discord bot that tells you if a song's lyrics are happy, sad, or neutral. You can type commands like /sentiment Post Malone + Circles and the bot will analyze the lyrics and make a graph showing how the mood changes throughout the song. The idea came from what I saw as a limitation in Project 1. In that project, I chose 6 specific artists. Everything in my code was built around those artists. If someone wanted to analyze a random artist they liked, they couldn't. My code wasn't flexible; it was hard coded around my choices in the three dead and three alive emo rappers I chose to analyze. I wanted to make it flexible and usable by anyone with discord (and of course the rest of the bot and api keys and helper files needed).

I also liked the idea of using Discord because it's where people already connect, share music, and talk about whatever. Instead of running a Python script alone on my laptop, I wanted some portfolio experience with a bot where multiple people could be like "Hey, what's the vibe of this song?" and see the results right away.

I started the project by creating a simple bot and spotify helper file for our very last participation assignment. I am SO glad I followed the recommendation to code something that would be useful for our final project as it got me working on the project well in advance when I otherwise probably would have procrastinated longer. I was also able to get peer feedback to apply to the project which I otherwise wouldn't usually have for a project aside from feedback on our proposals.

I decided to utilize helper files as that was Dr. Zietz's organizational style for the discord bot we made in class. I made four separate files: the spotify helper file, the genius helper file, the vader helper file, and the music bot file. The helper files all held the functions for my bots commands. During an office hours meeting with Dr. Zietz, he had talked about separating concerns in code. Separating concerns is a design principle where you organize code so that different parts handle different responsibilities: each piece focuses on one specific job. When my sentiment analysis wasn't working right, I only had to look in vaderHelper.py. When Spotify wasn't returning the right data, the problem was in spotifyHelper.py. That made debugging so much easier than having everything tangled together in one massive file.

I went with three APIs for the project in order to make it work. Spotify gave me the artist information and their top 10 most popular songs. Genius provided the actual lyrics, and I went with the genius lyrics documentation example code where it excludes remixes and removes tags like chorus and verse 1

because these would mess up the sentiment analysis. Vader is the tool that figures out if text is positive, negative, or neutral. It was designed for analyzing tweets, or other social media posts, which actually works okay for song lyrics since both use more like casual and informal language.

Each API was something I had seen before as we'd used them in the class for class coding/notes, participation assignments, and for project 1. I did get to use some new features that I found on the lyrics genius documentation which was super cool to me. There is a way to search for a song by lyrics. It essentially does a reverse search where you use a command that runs a function to search the genius search engine, which searches through their entire database of songs looking for matches. It'll return a dictionary with information about songs that contain those lyrics. Genius doesn't need it to be exact, and returns whatever it finds ranked by relevance. Once we have that, we can extract the song title and result from the nested dictionary. SO COOL! I spent hours just figuring out how to navigate the nested dictionaries various APIs returned, but the way we navigated that in class in order to get a spotify's user ID was super helpful in helping me understand Genius.

Dr. Zietz had given me feedback to not analyze whole songs at once. I am not sure if the idea came from something we did in class or was just intuitive but I decided to break them into chunks in order to see how sentiment changes throughout the song. This made total sense... a song with a devastating verse and an uplifting chorus might average to "neutral," hiding the emotional journey. I experimented with different chunk lengths. Five words... The scores jumped around wildly. "I love this" would be positive, then "song about my pain" would be negative. Twenty words... Emotional shifts that happened mid verse were getting blended together and kind of washed out. Ten words... Just right, sort of. Most rap lines are 8-15 words, so 10 word chunks roughly aligned with how lyrics are actually structured. But— I think overall lyrical analysis is pretty tough because so many things factor into it that even doing line by line might not give us a reliable sentiment overall. Nothing was super shocking in a scientific sense, but it did confirm that averaging a whole song can hide interesting emotional structure. The plots made that very visual.

In terms of fully understanding my own code, here's where I had to be really honest with myself. I extensively explained this in my docstrings comment as well as in collaboration agreement. Claude showed me this line: `compoundScores = [score['compound'] for score in chunkScores]`. And I stared at it. I kind of understood it was getting compound scores, but I needed to study the syntax further. In Project 1, I would have just said "okay cool" and moved on. This time, I asked: "Can you show me the longer way to write this?" and Claude showed me:

```
compoundScores = [] # Start with empty list
for score in chunkScores: # Go through each score
    compoundScores.append(score['compound']) # Add just the compound number
```

That I understood completely. It's a loop that goes through each score and pulls out just the compound number. The short version is just... compressed syntax for the same thing. I made myself write both versions in my code comments. The short version is what I used (because it's cleaner), but I could explain it and the long version to anyone who asked.

I built five commands for my bot:

1) /toptracks [artist]: This shows an artist's top 10 songs. It searches Spotify, gets the artist's ID, fetches their top tracks, sorts by popularity, and sends back a numbered list.

2) /lyrics [artist] + [song]: This retrieves full lyrics from Genius. The tricky part was handling Discord's 2000 character limit - some songs have really long lyrics! I had to split them into chunks and send multiple messages if this would ever come up, which we learned how to do in class when we were doing the reverb bot.

3) /searchlyrics [lyric snippet]: This function lets you look up songs on Genius by typing in a snippet of lyrics. It sends that snippet to genius.search_lyrics(), pulls out up to maxResults matches, which case we've set 5, and returns a list of (song_title, artist_name) pairs... otherwise it returns None if nothing is found.

4) /sentiment [artist] + [song]: This analyzes the overall mood... positive, negative, or neutral. It gets the lyrics, breaks them into 10 word chunks, analyzes each chunk, calculates the average, and returns something like "Overall: Neutral (Score: 0.034)".

5) /sentimentplot [artist] + [song]: This creates a graph showing emotional progression throughout the song. It does everything /sentiment does, then generates a line graph as a png and uploads it to Discord.

Making the visualization work was one of the trickier parts. Breaking it down step by step made it manageable: Extract the sentiment scores, create x-axis numbers (1, 2, 3, 4...), plot the line, add labels, save it, clear the plot so the next one doesn't overlap (using clf). As far as the legitimacy of my bot... well, Vader isn't perfect. It was made for social media, not poetry, so it misses sarcasm, irony, and most metaphors. Lyrics are only part of a whole song... a singer can sound sad while saying happy words. Hip-hop slang is unique... "bad" means "good," "sick" means "awesome", etc. A song about overcoming struggle uses negative words while conveying a positive message. Vader can't see that nuance. Granularity matters too, which Dr. Zietz tried to help us understand through breaking things up line by line but also by leveling our expectations on the truthiness of these sentiment analyses. Averaging a whole song hides the emotional journey. A song with a sad verse and happy chorus becomes "neutral," erasing the story.

Originally, I wanted database storage, machine learning models, Spotify audio feature comparisons, user accounts (lol), and interactive web dashboards. This all would have been cool but far beyond my ability and scope for this project. So I built what I could explain: API integration, data chunking, sentiment analysis, Discord commands, matplotlib graphs. Simple, but mine. Maybe for a passion project I'll add those features. If I continue this project, I'd add Spotify audio features (do sad lyrics match minor keys?), database storage for instant repeat requests, better sentiment analysis, and more text analysis like word frequency and rhyme schemes. Maybe even a way to just start playing the song. I know I saw something when I was reading docs around youtube links. But for now, I have something that works, something I understand, something people actually can use and seems like magic and that feels like success.

In the end, I successfully built a Discord bot that makes sentiment analysis interactive and accessible. By combining Spotify, Genius, and vader, the bot gives anyone real time insight into the emotional content of songs. The chunk-by-chunk analysis reveals emotional changes that whole song averages miss, and the Discord interface makes the tool easy to use.

Yes, there are clear limitations like vader's approximations, lack of musical context, and the challenges of slang and sarcasm. But the bot still achieves its core goal: it lets users explore and visualize song sentiment in a simple, hands on way.

More importantly, this project represents a different kind of success for me personally. I built something I understand. I used AI as a tutor instead of a crutch. I chose simplicity over complexity because I valued learning over impressing. And that feels like the kind of work I'm proud to show in a portfolio, not just something that "runs" but something I can actually explain.

MAI TAI | APP 4:13 PM
Want to learn about some music?

1. /toptracks [artist name] - Get top 10 tracks for any artist
2. /lyrics [artist] + [song title] - Get lyrics for a specific song
3. /searchlyrics [lyric snippet] - Search for songs by lyrics
4. /sentiment [artist] + [song title] - Analyze sentiment of song lyrics
5. /sentimentplot [artist] + [song title] - Visualize sentiment throughout song
6. /sayhello [names] - Say hello to the bot!

Katie 4:14 PM
/sentiment bon iver + halocene

MAI TAI | APP 4:14 PM
Analyzing sentiment for 'halocene' by bon iver...
Sentiment Analysis

Overall: **Neutral** (Score: 0.047)

Katie 4:14 PM
/sentimentplot bon iver + halocene

MAI TAI | APP 4:14 PM
Creating sentiment visualization for 'halocene' by bon iver...
Sentiment progression for 'halocene' by bon iver:

