Software Design Mini Project 3: Text Mining
Analyzing Song Lyrics by an Artist
Gaby Clarke
Spring 2016

overview

I used musixmatch's lyric database to compare the songs of a single artist in terms of the polarity of the language employed in the lyrics, in hope of creating a graphical representation of that artist's works.

implementation

My program is divided into two phases: data retrieval and data analysis.

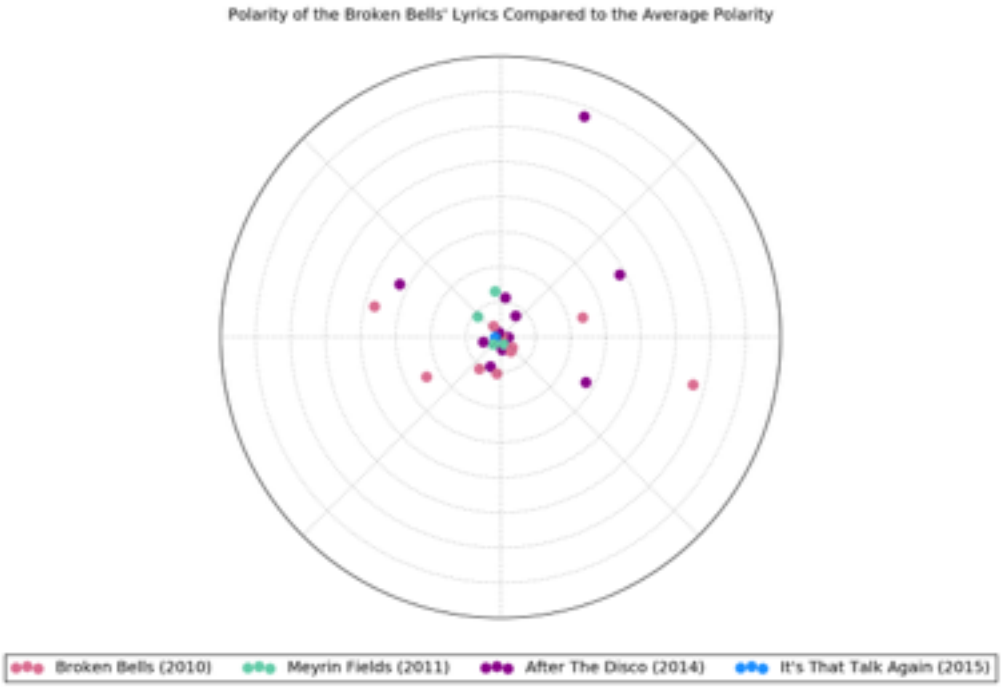
`lyricMining.py` retrieves data from musixmatch's database. The script does so by downloading the lyrics for all songs by a given artist (except remixes, which are redundant to the originals in lyric analysis), formatting the downloaded data, and saving the song lyrics to text files. The song list for a given artist is generated from the artist's home page on musixmatch, and the each song has its own page with its lyrics. The retrieval phase stores data in text files.

`lyricAnalysis.py` takes the formatted lyrics for each song and perform sentiment analysis on those lyrics using `Pattern`'s sentiment function. Specifically, the lyrics are analyzed for the polarity of the language. The polarity of each song is compared to the average polarity of the given artist's work, and that relationship is plotted using `matplotlib` on a polar axis, where the r-component of a song's position represents its difference from the average. The analysis phase largely employs dictionaries for data storage.

During the writing of this code, I made the decision to parse the plaintext for downloaded webpage html as opposed to the html, which, in hindsight, was probably the wrong decision. I chose to do so because I felt overwhelmed by what I was looking at, but had I chosen to instead parse the html, it would have been much easier to navigate to the individual song pages, as the links to those pages are embedded in the artist's page. I was also originally planning to use word frequency analysis to compare songs, but I ended up employing sentiment analysis instead because I found that it was more telling of the similarities and differences between songs.

results

I ran my program for the Broken Bells, one of my favorite bands. The analysis yielded the following results:



As you can see, the majority of the Broken Bells' songs fall in close to the average in terms of lyric polarity, so I decided to separate their repertoire by album and look for patterns there. The most interesting thing I found is that all of the songs from Meyrin Fields fall very close to the average. Meyrin Fields was an EP released after the release of the Broken Bells' debut self-titled album. The EP was composed entirely of unreleased tracks from Broken Bells… perhaps the band knew that these songs were *average* and as a result, decided they wouldn't make the cut for their first album? Probably not, but who knows. The one real outlier of the set, in the upper right, is Holding On For Life from After the Disco. The song has a relatively upbeat tone, and the lyrics fluctuate almost line by line between positive and negative, which is most likely the reason the song had a high polarity relative to the average.

reflection

For me, this project ended up being very much focused on obtaining data as opposed to analyzing data. I'm not sure if that was the intention, but I implemented my data analysis components rather quickly, and I didn't really feel like I learned that much from that implementation (or that the analysis yielded particularly compelling results). I do, however, feel like I learned a lot during the data retrieval phase. I definitely felt busy throughout the process, but not overwhelmed, so I'd say that the project was appropriately scoped. Regarding unit

testing, I found that most of the features I implemented didn't really have base cases or examples to test, the debugging was mostly dependent on me using actual data.

I would definitely like to do more with data visualization. I barely touched on the subject in this project, but it seems really cool.