**English Song Lyrics Crawler and Analysis Extension**

Tatiana Vinogradov (vingrdv2)

CS 410: Text Information Systems

Final Project

# 1. Overview

Music has become such an integral part of people’s everyday lives. We have seen the way it impacts lives on both the social and emotional level. Music can even send important messages and bring people together. And with the the growing number of music being released every day, so too grows the importance of collecting its data and developing an analysis of the topics and messages that today’s music brings across.

One of the best methods of doing this is through the analysis of song lyrics themselves, as they are generally the aspect of music that is most easily accessible. Even if given the copyright permissions to access an artist’s music, any audio and recordings may be difficult to translate into readable data that we can use. However, since lyrics are literally made up of text—which generally contain the topics and messages the artist is trying to get across—we can utilize text analysis tools to retrieve data and generate analyses based on their evaluations.

My project is an extension upon the Pop Song Lyrics Crawler, developed by previous students, Yuchen Lu and Zixin Huang. Their project had two purposes: 1) to generate lyrics data from an online database and create a well-structured dataset, and 2) make it clean and ready for further processing.

While the current implementation of their project is generally robust and well-structured, I could not help but notice that many improvements could be made. For example, their use of the Latent Dirichlet Allocation (LDA), using the MeTA Python library we used in class, can only read a small part of the dataset at a time, so I decided to implement a more robust and scalable version using the gensim Python library. Also, the former members expressed a desire that some improvements can also be made to the lyrics dataset they’ve generated, such as getting rid of redundant (i.e. “ohhh” or “yeaaah”) and abbreviated words.

The main improvements I brought to the project include:

1. Cleaning the dataset further to remove not only stopwords, but redundant and abbreviated wording.
2. Implementing a more scalable LDA package that can analyze a larger part of the song lyrics dataset.

# 2. Existing Systems

Plenty of online lyrics databases exist (i.e. MetroLyrics, MLDb, Lyrics.com). However, they only provide a way for the public to search and contribute lyrics online. There is no way to download and be able to use their data for further analysis. Also, because most of these websites support user-submitted lyrics, sometimes the lyrics are inaccurate, misspelled, or contain different formats (i.e. some have “[chorus]” to preface a song’s chorus while others do not). There are also already some song lyrics crawlers found on Github, such as this Genius Lyrics Crawler[[1]](#footnote-1) and this one that takes lyrics from songs on Spotify[[2]](#footnote-2). However, the latter only provides lyrics for the song you are currently playing without generating any data for it, and the former requires several different softwares to be downloaded and used, rather than an all-in-one software solution.

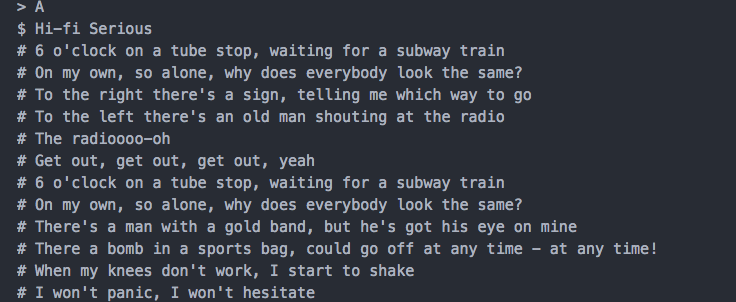
While certain crawlers may provide different levels of usefulness, depending on what the user is looking for, the difference with this Song Lyrics Crawler is that it generates a clean, well-structured dataset that users can then use and be able to form analyses based on this data. This data can also be easily downloaded, so that users and even researchers can use it for text analysis. It is easier to read and understand, as well as more adaptable for further work such as LDA analysis. Researchers need only download the specified Python packages, run the program, and they will have access to text files of song data for many different artists, which can also easily be adapted to only generate lyrics in different categories like genres, albums, or individual artists. LDA analysis can also be adapted and optimized for certain categories—say if a researcher only wanted to generate which topics one particular artist or group sings about the most.

# 3. Implementation

Since this is an extension of a previous project from over a year ago, it was important to first check that each current aspect of the project already implemented was up to date and still able to run. Some modifications therefore needed to be made to fit with current Python libraries. The lyrics database the previous members used, Lyrics123.net, has also changed and became SloLyrics.com. However, the overall format of the database remains the same, therefore the previous implementation that uses BeautifulSoup to parse the HTML pages need not be changed drastically. Also, while the previous members used Pickle to dump the data into text files before cleaning the data, I instead implemented my own functions to dump the artist, album, and lyrics information.

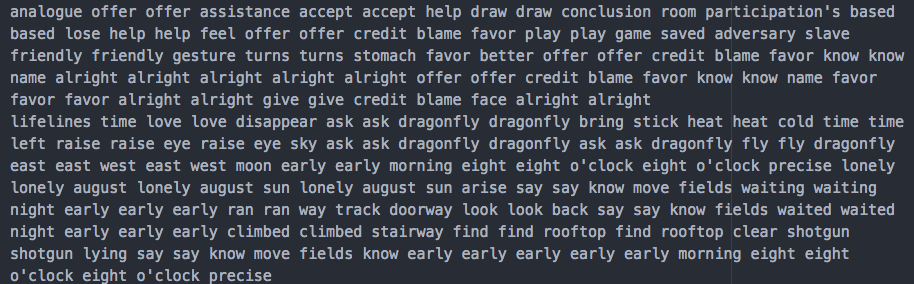
## 3.1 Parsing and Cleaning the Data

In crawler.py, BeautifulSoup and Python are used to parse the HTML of each artist’s webpage and save the artist, song, and lyrics information. Then, this information was dumped into separate text files organized by artists starting with the same first initial, which was then dumped into one large text file containing all the song data parsed.



A screenshot of the “songs.dat” file (artists are specified with > character, song title with $, and lyrics with #)

Once all the song lyrics were parsed from the database, the cleaning process began. Here I implemented my own function in remove\_words.py to lemmatize the text, remove stop-words, punctuation marks, and also check for redundant wording such as “ohhhh” and “yeaaah” that artists generally tend to include in their songs. To remove the redundancy, I used enchant to import an English dictionary that checks if the word is a common English word.



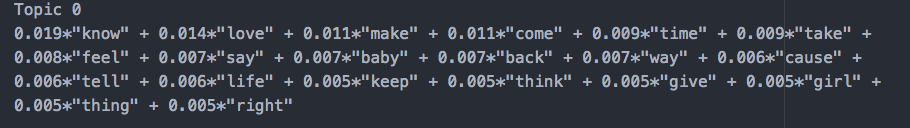
A screenshot of the now cleaned dataset that is ready to use for further analysis, found in “songs\_data.dat”

## 3.2 LDA Analysis

In the previous implementation, the members used MeTA for LDA topic modeling. However, MeTA is not very scalable and can only analyze part of the dataset at a time. This isn’t helpful as in order to create an accurate model of the topics songs are about, we need as much training data as possible.

To replace MeTA, I used gensim[[3]](#footnote-3) for LDA topic modeling. It is much more scalable, robust, and is specifically designed to handle large text collections using data streaming and efficient algorithms. It is also open source and therefore can be used by anyone. In order to lemmatize the data and create dictionaries, I also used NLTK.

This was found to be much more effective, as I was able to more efficiently generate topic models for all 38,272 songs that I parsed as opposed to the 1,783 songs the previous members could with MeTA.



Example of 1 topic with top 20 words displayed

# 4. Further Improvements

1. An even faster, more efficient LDA package that perhaps uses multicore processes and parallelization to cut down run-time substantially. Even with gensim, very large datasets can sometimes take up to hours to run, depending on the size of the dataset and the number of topics and iterations.
2. Being able to analyze the trend of song lyrics over time.
3. Topic analysis over different genres.
4. Separate analysis across different languages. This current implementation only analyzes songs in English to make it easier to weed out informal and abbreviated wording, but a lot more insight could be discovered if this type of lyrics analysis was spread across to different languages.

1. <https://github.com/floscha/genius-lyrics-crawler> [↑](#footnote-ref-1)
2. <https://github.com/willamesoares/lyrics-crawler> [↑](#footnote-ref-2)
3. <https://github.com/RaRe-Technologies/gensim> [↑](#footnote-ref-3)