GhostWriter: Project Final Report

Luke Novak, Nikhil Phatak, Michael Wheeler
CS4120: Natural Language Processing
Professor Lu Wang, Spring 2020

Abstract

We present *GhostWriter*, an application of natural language generation techniques to music and poetry. We created a dataset of song lyrics spanning many genres and periods by performing web scrapes of two major lyrics websites. We then used our dataset to fine-tune state-of-the-art transformer models like GPT-2 using HuggingFace's *Transformers* repository. Finally we evaluated performance by comparing perplexity values before and after fine-tuning, and prompted the model to produce novel lyrics resembling human output.

Problem Description

To ghostwrite is "to write (a speech, a book, etc.) for another who is the presumed or credited author." [1] Ghostwriting is an exciting application of natural language processing because it challenges commonly-held beliefs about art and creativity as domains exclusive to humans. Song lyrics form an unusual subset of natural language. Like most poetry, they often fail to conform to the syntactic and grammatical conventions of conversational language. What is more, lyrics tend to have additional structures such as rhyme and repetition absent from common language use. Predictive modeling can capture these features in either a supervised or unsupervised fashion; machine learning algorithms dealing with natural language can also process language on either a word-by-word or a character-by-character basis. Regardless, there are certain components common to all natural language generation systems; Table 1 outlines them as they pertain to song lyrics.

Table 1: Examples of the input and output of a lyrics generation system.

Input	Output
Algorithm or model architecture (BERT, GPT-2)	Trained or fine-tuned model containing learned coefficients
Labeled lyrics data (*"Do you hear me, do you feel me? We gon' be alright"* [2])	New generated lyrics (*"If you hear me I'll tell you all about my dreams you will be heartaches"*)
Prompt text (*"wide awake, eyes crossed by dream clouds"*)	Generated lyrics, using prompt text as context

Related Work

Table 2: A summary of the primary research we consulted in preparation for the project. Literature reviews and other secondary research sources are excluded.

Source	Dataset	Methods	Evaluation
[3]	40,000 songs	N-gram models	Rhyme frequency, matching syllable count
[4]	300 songs	Context-free grammars	Examination by human subjects
[5]	10,000 songs	Deep neural network	Rhyme density
[6]	200 verses	Long short-term memory model	Cosine similarity, rhyme density
[7]	100 songs	Naive Bayes, structure extraction	Cosine similarity, accuracy

We examined a handful of papers during the planning stage of our project, summarized in Table 2. Since the assignment did not require a formal literature review, we are unable to report a search methodology or specific inclusion criteria. Nonetheless the cursory searches we performed were useful in guiding our work on the project.

Our approach differs from existing work in the following ways. First, we created a new and much larger dataset, generated by performing a web scrape across Genius and Metrolyrics (see *Methodology*). We also made no attempt to filter by genre: our dataset contains all types of lyrics, while many previous attempts focused on a single genre like hip-hop. Finally, we used two different methods for generating new text and compared them intrinsically and extrinsically. Many of the above projects picked a single model type, and reported only one approach to evaluation of generated output.

Methodology

Dataset

We pulled lyric data for training from two sources: MetroLyrics and Genius. MetroLyrics describes itself as "the premiere destination for song lyrics online" and claims to provide the lyrics to over one milion songs. [8] Genius is a platform for community-driven annotation of music content, claiming to provide the "best lyrics and knowledge database on the internet." [9] Our current collection and preprocessing architecture consists of two scrapers followed by transformation steps to provide a consistent format. The MetroLyrics scraper leverages [10] to make API calls to MetroLyrics, using Billboard 200 data as a seed list for artists. The Genius scraper was built using [11] and uses a multi-tiered approach: first combing through an index of popular artists, then scraping the artist page for an API-specific artist ID, and finally using the artist ID to make API calls for all song pages to extract lyrics.

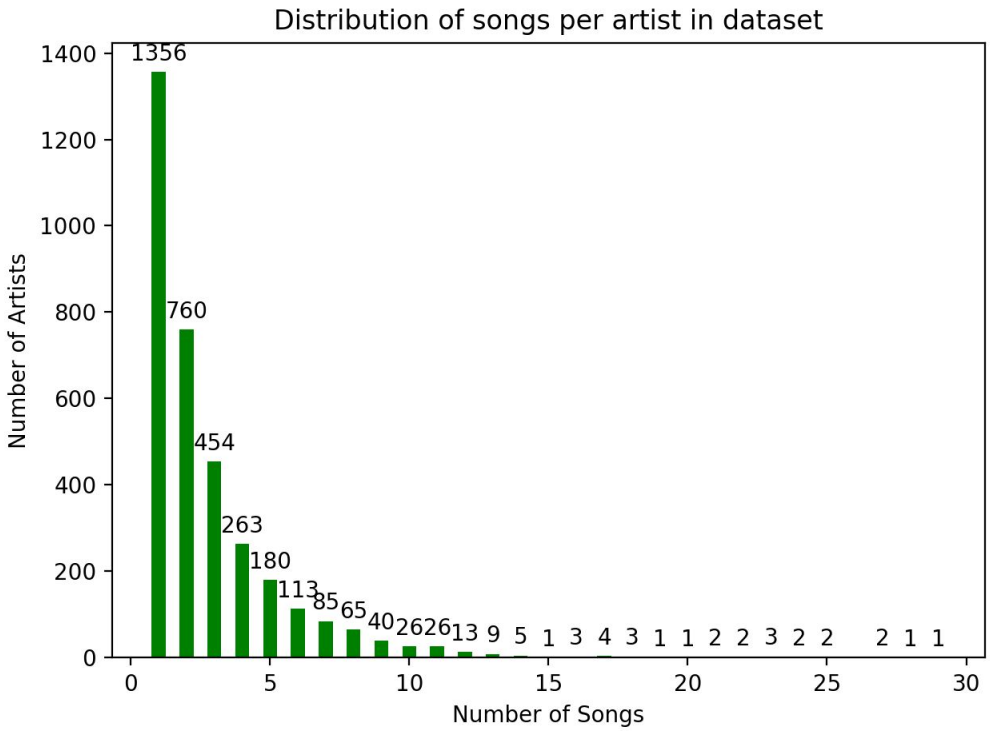

Figure 1: Distribution of songs and artists in the Metrolyrics dataset.

Ultimately, the use of the Metrolyrics-scraped dataset proved to be an accurate cross-section of American popular music over the last 50 years. The Billboard 200, which was used to determine what artists' songs should be accounted for in our dataset, is a list of the 200 most popular albums in the United States at a given time. As such, we were able to narrow our data down to mostly English songs without having to build a system to recognize the language of each song, and were guaranteed that the data we were using to train the model represented a relevant subset of music. This winnowing was not perfect, however, and we were still left with a number of songs in other languages, especially Spanish. Another benefit of using the Billboard 200 as seed data was that the size of the list (200 albums) was large enough to provide diversity of both artist and genre within the confines expressed earlier.

Upon collecting our dataset, we observed a number of promising traits that indicated that our dataset was sufficient for training and testing. First, as shown in Figure 1, the distribution of songs per artist indicated that we had works from thousands of different artists, and a selection of only a small number of songs. Outliers (one artist with 39 songs and one with 111) were omitted from the graph for the sake of visibility, however given that the vast majority of artists we sampled in the Metrolyrics database (slightly more than 3,000) had a low number of songs, we knew that we would not have data biased by the trends of a particular artist. This property of the dataset was also a drawback in that our desire to generate songs based on artist seed data would be difficult given the smaller sample size.

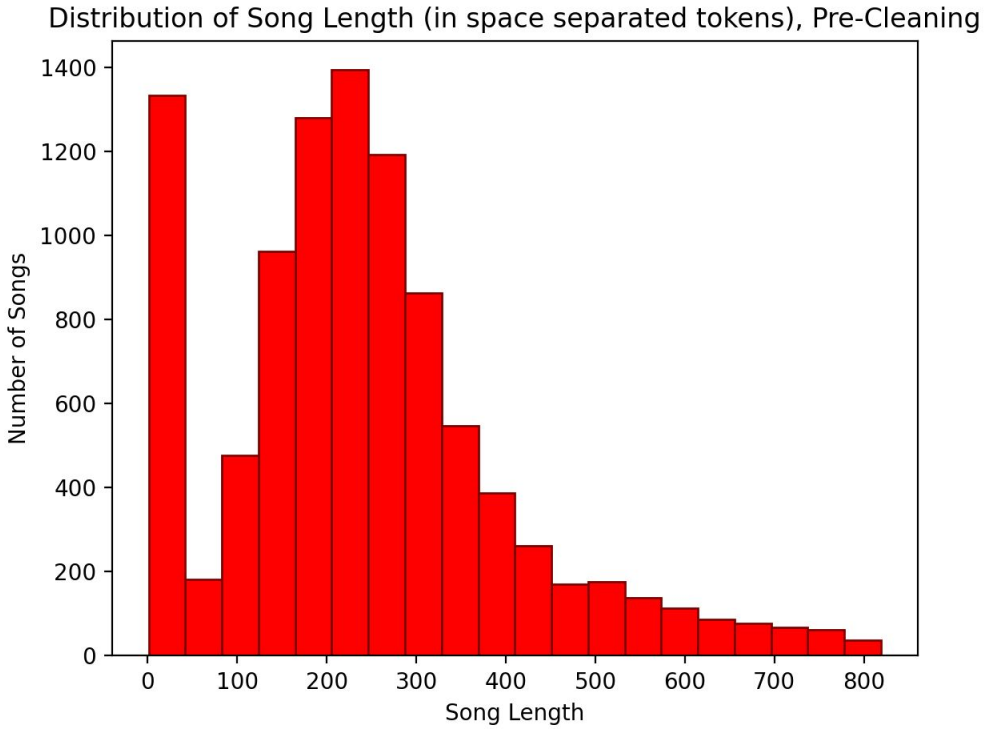

Figure 2: Distribution of song lengths before cleaning.[1]

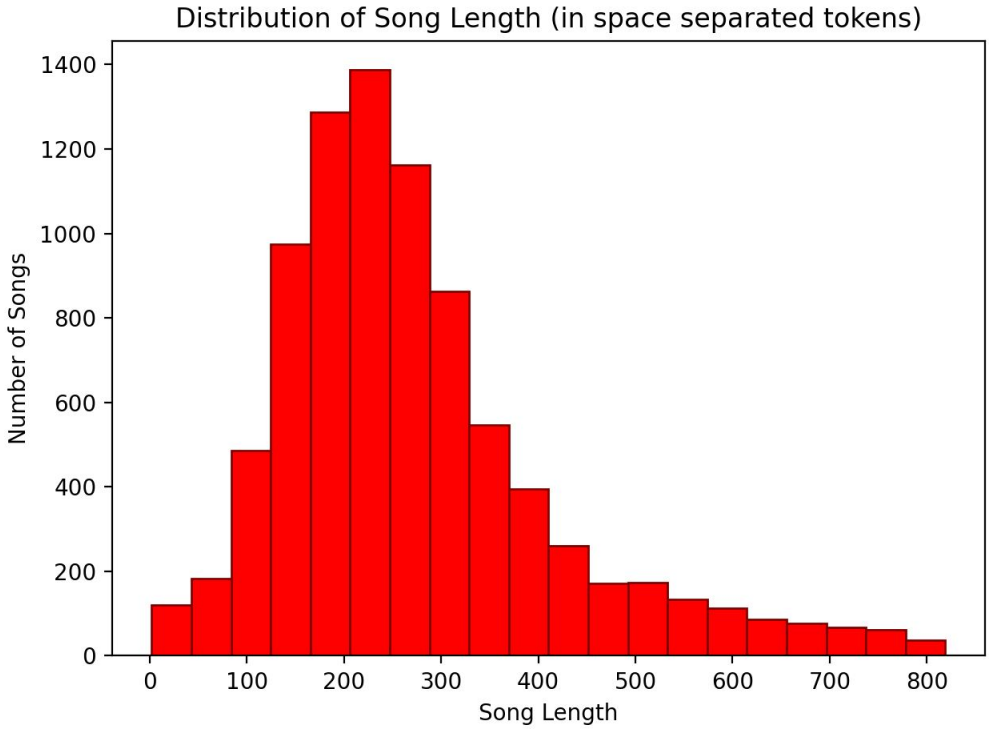

Figure 3: Distribution of song lengths after cleaning.[1]

[1] Top and bottom 1% excluded.

Another property we observed in our data was average length of a song, in space-separated tokens, which is shown in Figure 2. We have removed the top 1% of song lengths as outliers so that the distribution of lengths is more visible. As can be seen in Figure 2, song lengths look to be roughly in a normal distribution, with a large peak at the left end of the graph. When we looked at the song-level data to see why this peak existed, we noticed a number of songs without lyrics that were instead tagged with a single word, like "Instrumental" to denote that the song did not have any lyrics. Once noticed, we decided to exclude these songs from the data as they did not constitute actual song lyrics. With these songs removed, a fairly distinct normal distribution can be observed in Figure 3, with most songs falling between 95 and 400 tokens in length.

One other consideration was the diversity of words, as song lyrics can tend to be repetitive, especially when a song has multiple choruses with the exact same contents. After counting the individual unigrams across the Metrolyrics dataset, we determined that the data contained 2,434,629 words, of which the max word probability was about .039. The median word probability was 4.1e-07. Considering the highest probability of words in general English is around 0.060 [12], our analysis showed that our data set had sufficient diversity of words despite the repetition inherent in songs.

As of the time of writing we have collected a total of 500MB of lyrics text from over 100,000 songs between both datasets across a diverse array of artists and genres. Our preprocessing steps currently include stripping out all newlines and HTML tags, as well as annotating the lyric text with an artist ID for targeted downstream generation. This has so far allowed us to generate somewhat workable training data for end-to-end validation of our concept and process. However there remains work to be done to refine the dataset for higher quality and better learning (see *Future Work)*.

Preprocessing

The features we used were those leveraged by the models in [13]: word vectors and average vectors over phrases consisting of several words. In order to augment this, for our *Histories* based approach (see *Experiments*), we used the average of the previous *n* words as another feature for generation. In our implementation, *n* was fine-tuned to be 5.

Experiments

Using code available in our project repository [14] we produced specialized generated text using two methods:

- *Transfer learning on an existing generative language model:* This is done by freezing values in a pretrained model (BERT, GPT-2) and adding layers at the end of the network. Then, we run backpropagation using a specialized dataset (in our case, lyrics) on the extended network.

- *Histories:* Store a history of documents for each feature (genre, artist) and pass that history in at evaluation time for constructing the input we pass to the generative model (passing in a history is a built-in functionality of generative models provided by [13]).

We found that the *histories* approach did not produce particularly intelligible output, despite its perplexity (as seen in the Results section) not being much higher than the perplexity of the

transfer-learning-trained model. However, the perplexity was still higher, so we have deemed the *transfer learning* approach to be our favored approach.

This approach was made possible by the number of pre-trained models provided within the HuggingFace transformers library, which allowed us to leverage the training already done within these models, specifically GPT-2, and fine-tune it to our specific use case

Evaluation Metrics

We performed both intrinsic and extrinsic evaluation of our results. For intrinsic evaluation, we computed the perplexity of our fine-tuned GPT-2 model on different datasets and compared the results. This comparison tells us how well our transfer learning training worked - the lower the perplexity of the dataset, the more easily the model was able to predict the next word in a phrase. We also evaluated extrinsically, by passing our output through a text-to-speech service and playing generic music over it so we could try to hear any patterns or musicality that emerged (we exhibited one of these samples during our presentation).

Results

A noticeable difference between our two generative language model approaches is that the *transfer learning* approach evaluates much faster than the *histories* approach. After some investigation, we determined that this is most likely because we have extended the size of the allowable stored history to fit a large number of documents that score highly using a *TF-IDF* measure. Additionally, the *transfer learning* approach evaluates with a significantly lower perplexity than the *histories* approach, as is evident in the table below. Another significant result is that compared to the baseline, the *transfer learning* approach has a much lower perplexity when evaluated on the MetroLyrics dataset. This indicates that our fine-tuned model is much more predictive of song lyric syntax and semantic patterns than on regular text.

Table 3: Experiments on GPT-2 model from [13], and the resulting perplexity values.

Model	Dataset	Perplexity
Fine-Tuned GPT-2	Generic text with same number of tokens as dataset (baseline)	20.20
Fine-Tuned GPT-2	MetroLyrics (10,000 songs)	8.75
Pretrained GPT-2 with History Input	Generic text with same number of tokens as dataset (baseline)	45.43
Pretrained GPT-2 with History Input	MetroLyrics (10,000 songs)	38.90

Although we cannot empirically evaluate the generated lyrics, we wanted to include some examples to show the coherence of our output relative to the prompt text we input. The prompt texts are phrases from existing songs. In a production system, we envision a user would pick an artist or genre, and our system would randomly sample from a database of phrases that have those characteristics to use as the prompt text to pass to the generative model.

Table 4: Examples of text generated from a GPT-2 model fine-tuned on MetroLyrics data.

Prompt Text	Generated Output
"High as a pretty star, don't you break my faded heart" [15]	"Long legs wide awake eyes crossed by dream clouds If you hear me I'll tell you all about my dreams you will be heartaches If you're with me I'll go where you say you want And sleep my dreams I'll scream you a thousand times Cause there's always the chance of those nights out of sight"
"Hello darkness my old friend" [16]	"I shall not be weak Yet I think I will be the one to begin And if my soul dies Till then I will lay down My life is free I can't find you but by now I used to be to people I told, in the afternoon That I had left away Here, where I would listen to your music"
"I'm ready to party, let my hair down and have a good time"	"All I need is some sleep, you can find me somewhere safe and soft Your love made me wild, I found another way to live Are you not a mean girl Don't you love me?"

Table 5: Examples of text generated from GPT-2 pretrained (not fine-tuned), passing in MetroLyrics data as "history" to the model.

Prompt Text	Generated Output
"High as a pretty star, don't you break my faded heart" [15]	"no, but that's what she does to me. she just takes my money and tells me you are a real man, i can not do that, i am gonna find out soon, i'm a real man, i am gonna find out how far she's going to that's a good thing, i'm gonna find out how far she's going that is the spirit, you should look into it"
"Hello darkness my old friend" [16]	"hello darkness and i'm so tired of asking questions i'm so tired of asking questions i can help you sleep i can sleep in my old room i can help you sleep in my bed"
"I'm ready to party, let my hair down and have a good time"	"i am ready to go party, my friends are coming and i want to be a famous famous girl you are funny! i'm ready to go home i am ready to go party, so that is good i am ready to party i am ready to go party"

Future Work

While the quantity of lyrics data collected is promising, more effort spent on data cleaning would maximize the quality of future results. Since our data came from the internet a sizable fraction of the songs in our corpus are in languages other than English. Future preprocessing could involve applying a language recognition model such as [17] to exclude non-English song lyrics. There is also a fair amount of markup within the lyrics: text describing song structure like verses, choruses, and bridges. A set of regular expressions targeting square brackets and musical keywords may be effective to remove the meta-text. Finally, coalescing the artist ID tags between MetroLyrics and Genius would ensure that duplicate songs under the same artists are filtered out in order to leverage the entirety of our corpus together.

Another potential enhancement for the dataset is classification by genre to open up more modeling possibilities. Such an approach may allow for better generalization than tagging by individual artists with comparatively small discographies. To this end we could incorporate new datasets into the pipeline; a strong candidate would be [18]: leveraging their API would allow classification of artists by genre, and by extension their songs.

Finally, while we can extrinsically evaluate individual generated lyrics and intrinsically evaluate the model's holistic performance, it would be useful to have a metric of intrinsic evaluation of generated text. Past work points to adversarial evaluation as a means to this end. [19] However the complexity of training increases greatly with such an approach, necessitating the construction of a separate model with additional training data to discriminate between human and machine generated responses.

Acknowledgements

Thank you to the creators and maintainers of all the HuggingFace repositories, as well as the researchers at OpenAI who produced the research that led to GPT-2. Thank you also to Prof. Lu Wang and Akshay Dangare for their feedback on the project proposal and the expertise they have provided throughout the course.

References

[1] "Definition of GHOSTWRITE." https://www.merriam-webster.com/dictionary/ghostwrite (accessed Apr. 19, 2020).
[2] "Kendrick Lamar – Alright." https://genius.com/Kendrick-lamar-alright-lyrics (accessed Apr. 19, 2020).
[3] H. Nguyen and B. Sa, "Rap Lyric Generator," Stanford University, Jun. 2009. Accessed: Jan. 26, 2020. [Online]. Available: https://nlp.stanford.edu/courses/cs224n/2009/fp/5.pdf.
[4] S. Pudaruth, S. Amourdon, and J. Anseline, "Automated generation of song lyrics using CFGs," in *2014 Seventh International Conference on Contemporary Computing (IC3)*, Aug. 2014, pp. 613–616, doi: 10.1109/IC3.2014.6897243.
[5] E. Malmi, P. Takala, H. Toivonen, T. Raiko, and A. Gionis, "DopeLearning: A Computational Approach to Rap Lyrics Generation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2016, pp. 195–204, doi: 10.1145/2939672.2939679.
[6] P. Potash, A. Romanov, and A. Rumshisky, "GhostWriter: Using an LSTM for Automatic Rap Lyric Generation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, Sep. 2015, pp. 1919–1924, doi: 10.18653/v1/D15-1221.
[7] J. P. G. Mahedero, Á. MartÍnez, P. Cano, M. Koppenberger, and F. Gouyon, "Natural Language Processing of Lyrics," in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, New York, NY, USA, 2005, pp. 475–478, doi: 10.1145/1101149.1101255.
[8] "About us | MetroLyrics." https://www.metrolyrics.com/about.html (accessed Mar. 23, 2020).
[9] "Genius – About Genius," *Genius*. https://genius.com/Genius-about-genius-annotated (accessed Mar. 23, 2020).
[10] S. Brennan, *tswift: MetroLyrics API*. .
[11] "Scrapy | A Fast and Powerful Scraping and Web Crawling Framework." https://scrapy.org/ (accessed Mar. 23, 2020).
[12] "Distribution of Words - Stopwords." http://faculty.georgetown.edu/wilsong/IR/WD3.html (accessed Apr. 19, 2020).
[13] *huggingface/transformers*. Hugging Face, 2020.
[14] L. Novak, N. Phatak, and M. Wheeler, "lukenovak/ghostwriter," *GitHub*, Apr. 19, 2020. https://github.com/lukenovak/ghostwriter (accessed Apr. 19, 2020).
[15] "BØRNS – Faded Heart." https://genius.com/Brns-faded-heart-lyrics (accessed Apr. 19, 2020).
[16] "Simon & Garfunkel – The Sound of Silence."

https://genius.com/Simon-and-garfunkel-the-sound-of-silence-lyrics (accessed Apr. 19, 2020).
[17] M. M. Danilak, *langdetect: Language detection library ported from Google's language-detection.* .
[18] "Web API | Spotify for Developers." https://developer.spotify.com/documentation/web-api/ (accessed Mar. 12, 2020).
[19] D. Jurafsky and J. H. Martin, "Chapter 26: Dialogue Systems and Chatbots," in *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed., Upper Saddle River, New Jersey: Pearson Education, Inc., 2019, p. 12.