Deep Learning in Musical Lyric Generation: An LSTM-Based Approach

Harrison Gill[1], Daniel (Taesoo) Lee[1], Nick Marwell[1]

[1]*Department of Linguistics, Yale University*

Abstract

This paper explores the capability of deep learning to generate lyrics for a designated musical genre. Previous research in the field of computational linguistics has focused on lyric generation for specific genres, limited to Recurrent Neural Networks (RNN) or Gated Recurrent Units (GRU). Instead, we employ a Long Short Term Memory (LSTM) network to produce lyrics for a specific genre given an input sample lyric. In addition, we evaluate our generated lyrics via several linguistic metrics and compare these metrics to those of other genres and to the training set to assess linguistic similarities, differences, and the performance of our network in generating semantically similar lyrics to corresponding genres. We find our LSTM model to generate both rap and pop lyrics well, capturing average line length, and in-song and across-genre word variation very closely to the text it was trained upon.

1. MOTIVATION

Neural network-based language models have shown promise in producing original long-form prose content from minimal initial text. This work has been further extended to attempt musical lyric generation. Language modeling for lyrics, however, poses several challenges which normal prose does not. Modeling line breaks is critical to success, stylistic elements such as flow, rhyming, and repetition are staples of the best lyrics, and lyrical structures such as verse-chorus form are critical to producing "good" songs. While these attempts have resulted in varying degrees of success, it has not been acknowledged that "lyrics" itself is a broad category. Different genres of music are notably different in lyrical style, which is reflected in linguistic features including but not limited to line length, word repetition, word variation (both within songs and within genres as a whole), semantics, and the propensity to write in the first, second, or third person. Our work explores the ability of neural networks to generate genre-specific song lyrics that preserve the aforementioned linguistic features native to each genre.

2. RELATED WORK

Lyrical generation via LSTMs has been explored for specific genres. Potash et. al (2015) in *GhostWriter: Using an LSTM for Automatic Rap Lyric Generation* attempted to synthesize lyrics as a "ghostwriter," a creator of lyrics for a specific artist. However, their model was limited in generating lyrics for a genre, as it was trained on a specific artist. Additionally, Potash et. al implemented rhyming by training their model with sets of lyrics in which corresponding rhyming words were noted.

Watanabe et. al (2018) explored in *A Melody-conditioned Lyrics Language Model* the creation of a model that produces entire lyrics for a given input melody. Pairing lyrics and the corresponding melody allowed them to perform melody-condition lyric generation, achieving impressive results. However, they use a well-explored RNN-based language model, rather than utilizing deep LSTM architectures that have achieved state-of-the-art performance on language model tasks.

There exists potential to expand both Potash et. al and Watanabe et. al's research to have full genre capabilities. As is evident with our own results and the analysis of these other lyric generation papers, datasets with the proper supplemental characteristics (melody identification, rhyme identification) is essential in producing lyrics that closely resemble songs for the given artist or genre. However, we take the promising LSTM-architecture and apply it across multiple genres and artists, expanding both the scope of lyrical generation via contemporary deep-learning techniques.

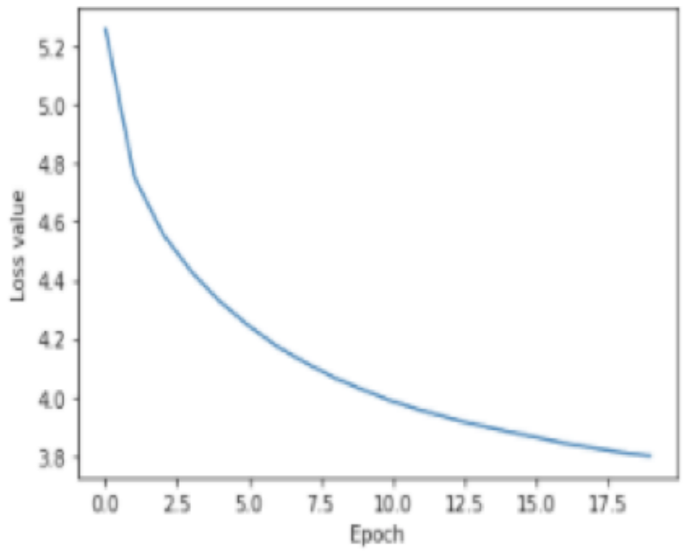
3. APPROACH

We created a dictionary of words based on the lyrics sampled from the corpus of songs for a given genre. The premise of our network is that it would take in a string of **k** words, and output the next word, newline character, or punctuation in the lyrics. The network then takes the 2-**k** words from the original input string, concatenates them with the newly predicted word, and uses that as the next input to the model. After some tuning, we arrived at 16 as the input size (the value of **k**). The decision to set **k** equal to 16 was the result of clear tradeoffs between compute and complexity. Greater values of **k** tended to produce better models (all else held constant) but increased computational cost, forcing us to shrink the size of training datasets, dimensionality of layers, or number of training epochs. 16 was the smallest value at which our network started picking up many of the structural long-term dependencies in lyrics, most notably the verse-chorus form, which was reflected in many of the songs it produced. It was not always large enough, which was obvious from the number of times songs would seemingly switch topics part way through, but it usually got the job done. Among other things, smaller values of **k** were particularly susceptible to reaching a point where they started repeating the same line over and over again. We also decided to focus on only six genres due to computational resources: Rock, Pop, Rap, Metal, Country, and Jazz.

4. NETWORK ARCHITECTURE

In previous research and literature discussed in earlier sections, LSTMs have proven to be promising for text generation, more specifically lyric generation. Our network's first layer is an embedding layer, which importantly is not pretrained. After the embedding layer, we use an LSTM layer with dropout, a regularization method in which input and recurrent connections to LSTM units are excluded from activation and weight updates to reduce overfitting. Finally, we use a linear layer to output a vector the length of the vocabulary as the output layer. Softmax is applied to convert this to a vector of probabilities.

Figure 1

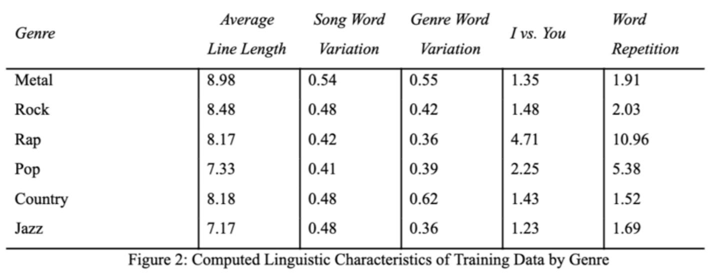


No elements were included in our network to force good lyric-generating behavior. A good example of this is that there is no non-neural network piece of our code to force rhyming as has been implemented in previous lyric-generating language models. The reason for this is that our goal was less to produce outstanding lyrics and more to observe what neural networks could and could not capture. Our lyrics rarely rhymed, but that informed us that these networks could not pick up on rhyming.

5. TESTING IMPLEMENTATION

We acquired data from two sources. Initially, we used Kaggle's *380,000+ lyrics from MetroLyrics* dataset, which featured thousands of songs with their lyrics from the following genres: Rap, Rock & Roll, R&B, Indie, Country, Jazz, and Other. This dataset did not provide sufficient songs for some genres, provided incomprehensive breadth in other genres (i.e. only including artists up to "G" alphabetically), and heavily consisted of lyrics composed of foreign languages. In response to this, we decided to build a web scraper to pull lyrics via the *Geniuslyrics* API (Genius 2020), the world's biggest collection of song lyrics. After doing so, and after cleaning the dataset (such as removing non-English songs, songs with unspecified genres, and instrumental songs), the compiled dataset included 297,876 songs, each labelled one of 14 genres. This complete dataset contained more than 2,000 songs for each genre, with rock having the plurality of songs, with 109,221. Figure 2 describes the studied linguistic characteristics of the 6 studied genres: rap, rock, jazz, country, metal, and pop. We devise five metrics below for evaluating the lyrics in our training dataset, along with the lyrics that we produce.

Figure 2: Computed Linguistic Characteristics of Training Data by Genre



Genre	*Average Line Length*	*Song Word Variation*	*Genre Word Variation*	*I vs. You*	*Word Repetition*
Metal	8.98	0.54	0.55	1.35	1.91
Rock	8.48	0.48	0.42	1.48	2.03
Rap	8.17	0.42	0.36	4.71	10.96
Pop	7.33	0.41	0.39	2.25	5.38
Country	8.18	0.48	0.62	1.43	1.52
Jazz	7.17	0.48	0.36	1.23	1.69

Figure 2: Computed Linguistic Characteristics of Training Data by Genre

1. ***Average Line Length*** – number of words in each line of a lyric
2. ***Song Word Variation*** - number of unique words, normalized (divided by the total number of words in a lyric)
3. ***Genre Word Variation*** - number of unique words in the generated songs for that genre
4. ***I vs. You (Point-of-View)*** **-** number of lines that started with an "I" and subtracted the count of lines that started with "you". By doing so, we measured/captured the point-of-view of the lyrics
5. ***Word Repetition*** - number of occurrences of a repeated word. For example, "please talk talk to me" would count as a single Word Repetition, whereas "you make me feel good good good" would count as two Word Repetitions.

Aside from the rudimentary metrics above that assess word variation (1-3), we chose to focus on Point-of-View and Word Repetition. A lyric's point-of-view conveys perspective through which the song is sung and depicts the relationship between the artist and the listeners, which is a core aspect of what separates listeners across genres. For example, it is known that a significant portion of rappers employ the first person point-of-view in their lyrics as a form of expression. In addition, word repetition may be more common in certain genres characterized by high counts of alliteration, such as rap, when compared to other genres. Lastly, following Malmi (2016), we did consider including rhyme density as a 6th metric, but concluded that this metric would not vary as much for non-rap genres, as it is known that formal structure of rhyming is present in rap lyrics and provides flow to the music whereas this is not well documented in other genres.

In Fig. 2, we observe that metal has the longest average line length. The greatest amount of in-song word variation occurs within metal as well, but the country genre captures the largest amount of in-genre word variation. In addition, rap seems to have more lyrics in the first person point-of-view, along with more word repetition. The latter may be attributed to short-syllable and alliteration techniques used commonly in rap music.

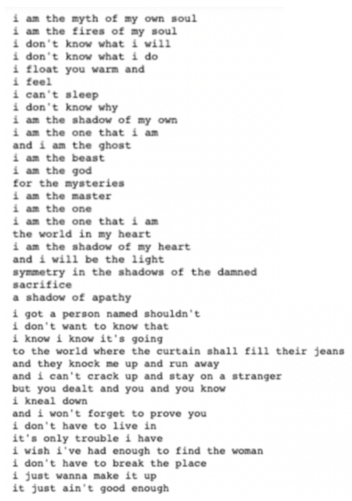
6. ANALYSIS TECHNIQUES

Despite having 14 genres, we narrowed in on jazz, country, metal, pop, rap, and rock genres given the semantic difference between their lyrics (i.e. rap and hip-hop are fairly similar styles of music) and the limit of computational resources to which we had access. For each of these genres, a model was trained on 2000 sampled songs from our dataset. We then created 20 different song “prompts,” each of length 16, and fed those same song “prompts” into each of the models trained on each of the genres. Several of the prompts are listed below:

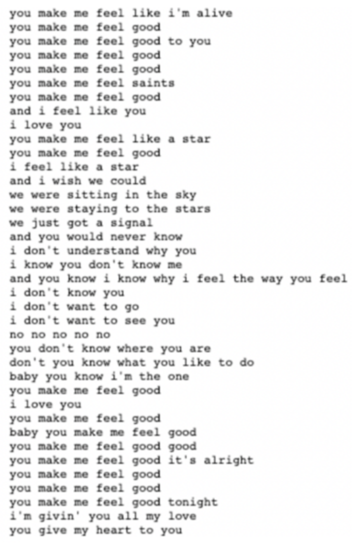
- *“I like long walks on the beach \n and shopping sprees in paris \n I sometimes”*
- *“It was a dark day when he died \n my mom had tears in her eyes”*
- *“Yeah \n yeah \n yeah \n this is the way I talk when I’m mad \n”*
- *“Drive forever \n baby I need your heart \n free car \n just another a wild.*
- *“The way that you love me \n is hard to explain \n I’m addicted to your”*

We also analyzed the ability of the model to generate lyrics that qualitatively (and semantically) resembled lyrics of the genre on which the model was trained. With the exception of almost never rhyming, these models fared shockingly well. They almost never stuck to the original sixteen-word prompt, but the songs were evaluated only on what was generated by the model (not including the original prompt). The lyrics felt semantically apt in their relevant genre (see the metal example in Figure 3), and often took on real lyrical flows and structures (see the production of verse-chorus form in the pop song of Figure 3).

Figure 3



```
i am the myth of my own soul
i am the fires of my soul
i don't know what i will
i don't know what i do
i float you warm and
i feel
i can't sleep
i don't know why
i am the shadow of my own
i am the one that i am
and i am the ghost
i am the beast
i am the god
for the mysteries
i am the master
i am the one
i am the one that i am
the world in my heart
i am the shadow of my heart
and i will be the light
symmetry in the shadows of the damned
sacrifice
a shadow of apathy
i got a person named shouldn't
i don't want to know that
i know i know it's going
to the world where the curtain shall fill their jeans
and they knock me up and run away
and i can't crack up and stay on a stranger
but you dealt and you and you know
i kneal down
and i won't forget to prove you
i don't have to live in
it's only trouble i have
i wish i've had enough to find the woman
i don't have to break the place
i just wanna make it up
it just ain't good enough
```



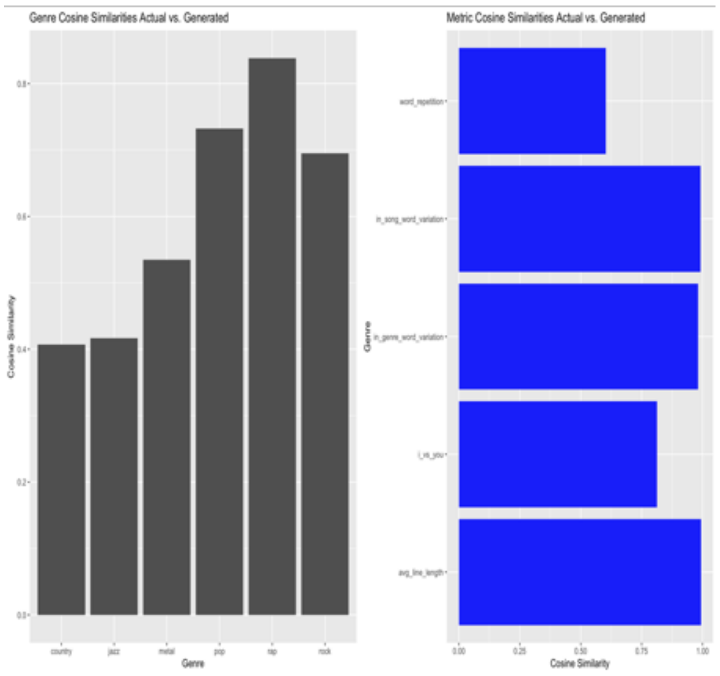
```
you make me feel like i'm alive
you make me feel good
you make me feel good to you
you make me feel good
you make me feel good
you make me feel saints
you make me feel good
and i feel like you
i love you
you make me feel like a star
you make me feel good
i feel like a star
and i wish we could
we were sitting in the sky
we were staying to the stars
we just got a signal
and you would never know
i don't understand why you
i know you don't know me
and you know i know why i feel the way you feel
i don't know you
i don't want to go
i don't want to see you
no no no no no
you don't know where you are
don't you know what you like to do
baby you know i'm the one
you make me feel good
i love you
you make me feel good
baby you make me feel good
you make me feel good good
you make me feel good it's alright
you make me feel good
you make me feel good
you make me feel good tonight
i'm givin' you all my love
you give my heart to you
```

Cosine similarity was used to evaluate these five metrics in two ways. First, the cosine similarity between the metric vectors for original and generated songs of each genre. For example, the cosine similarity between v_1 and v_2, where v_1 is a vector containing the value of each of the five metrics for a genre with our generated lyrics and v_2 is a vector containing the value of each of the five metrics for a genre for the original lyrics from our dataset. Second, the cosine similarity between the metric vectors for original and generated songs by metric. For example, the cosine similarity between w_1 and w_2, where w_1 is the vector of values for a single metric measured across different genres from our generated lyrics and w_2 is the vector of values for a single metric measured across different genres from the original lyrics.

7. RESULTS

After calculating these metrics for the lyrics corpus on which we trained each individual model, we computed those same metrics for the generated text to determine if our model was able to generate semantic/linguistic similarities. To compare the original text to the generated text, we computed cosine similarities, as shown below in Figure 4.

Figure 4



From the left graph of Figure 4, it is clear that the metrics computed on generated lyrics were the closest to the original lyrics in the rap genre, followed closely by pop and by rock. It is important to note that although the magnitude of our metric vectors for the generated text vs. the actual text was different, the model does preserve relative values between genres for a given lyric. An example of our rap generated lyrics is shown below, in Figure 5.

The graph on the right in Figure 4 represents which of the five metrics our model was able to pick up the best across all genres: average line length, and in-song and across-genre word variation. We were pleasantly surprised by the ability of our model to pick up on cosine similarities above 0.5 for all our metrics. Figure 6 presents a table showing the percentage changes in each metric moving from the original song lyrics to the generated song lyrics in a given genre. Here, we do observe a decent amount of variation before and after song generation, most notably how our model is unable to capture our *I vs. You* metric in rock.

8. ALTERNATIVE METHODS AND FUTURE RESEARCH

Initially, we had intended on creating character-level LSTMs. Given that we were using 16-word windows to generate lyrics, any character-size window would have required significantly more computational power to predict words, character by character. Additionally, we hypothesized that generating lyrics at a word-level would allow us to pick up on semantics within the text, such as rhyming, chorus generation, and other word-based properties. For these aforementioned reasons, we stuck to a word model.

We explored both BERT and GPT-2, two transformers, to experiment with lyric generation. It was rather apparent immediately that text generation via BERT would be next to impossible due to its bidirectional nature—BERT works best when it is predicting a word given context words on both sides, but in our case context was only provided prior to the word being predicted, so BERT's bidirectionality, a lynchpin of its architecture, was rendered useless. When attempting lyric generation with GPT-2, we obtained mediocre results. The core problem with these GPT-2 results is the non-musicality of the resulting generated lyrics. GPT-2 was trained on a large corpora of prose-like text, which is fairly different from how lyrical text is structured. Prose does not care about the same rules as musical lyrics. Line lengths are often dictated by the physical restrictions of the page, not the creative intent of the author; rhyming in prose is infrequent at best, and word repetition is typically frowned upon, not encouraged. The process of starting from a large pre-trained model and adapting it to a specific task is a battle to break certain habits of the large general model, and emphasize or even introduce good ones. The more the large pre-trained model has seen a certain specific behavior, the deeper it is engrained in the model and the harder it is to change that behavior. It turns out that prose style is deeply ingrained in GPT-2. Because of its prose-like upbringing, we lacked the computational power to alter the nature of GPT-2 from prose to lyrical, and hence it failed to be an apt model for lyric generation, despite often being the most coherent in content and grammar. In short, the GPT-2 lyric generator lacked flow. Examples of this meaningful difference include the propensity for words which rhyme to appear in certain relationships with one another, the use of newline characters, and the propensity for certain words to be repeated. While the content produced by pre-trained embeddings made sense grammatically and coherently, lyrics generated with pre-trained embeddings did not feel like lyrics at all.

Future avenues of work consist of employing different types of models to our song lyric data, most notably simple Markov Models (as those are the simplest way to predict the next word using previous words via transition probabilities) as customization can be done due to the simplistic nature of the model. Secondly, GRUs may lend to quicker training given fewer gates than LSTM, but it remains unclear whether performance would be maintained. Lastly, we may produce interesting results if we allow each model to train on all the data, not a singular genre, but weigh the relevant genres more than the rest in training. The reasoning behind this is the model could possibly assign different weights to genres and how they should be used for encoding/training—it is a known fact that genres do have overlap in their linguistic structure, for example, as rap may resemble certain parts of pop.

Figure 5: Percentage Changes from original to generated lyrics for each metric and genre

Genre	*Average Line Length*	*Song Word Variation*	*Genre Word Variation*	*I vs. You*	*Word Repetition*
Metal	-54.3	-25.1	-13.5	82.7	88.2
Rock	-38.4	-5.0	-5.9	77.0	34.9
Rap	-52.3	-64.6	-99.0	75.5	-49.6
Pop	-28.0	-16.7	-44.6	94.6	36.5
Country	-74.1	-36.6	-23.0	-229.6	75.2
Jazz	-24.1	-38.7	-78.5	73.7	94.6

It is important to acknowledge that given computational limitations of our work, the ability of pre-trained models to adapt to different linguistic styles (i.e. prose vs. lyrical as explored here) may prove possible given more compute and hence the ability to leverage a larger corpora. We hope others will further explore this idea in musical lyric generation and beyond to see if general models trained almost exclusively on prose can be effectively adapted to the writing styles of non-prose formats (poetry, note-taking, and speeches are a few examples). Within the broader field, we also wish for this to provide a jumping off point for more personalized text generation. While classification between different writers is a well-documented task in NLP, switching that task from classification to generation, and asking to move up a level (from individual writers to those who group themselves together by genre within a certain writing style) is much less explored. It is a fundamentally important question as writing is often meant to be a targeted, audience-in-mind, exercise. If our target audience is hip-hop aficionados, we shouldn't write the same lyrics as we would for jazz masters, and this paper shows that generation models can capture these differences not just semantically, but in style as well.

ACKNOWLEDGEMENTS

We would like to thank Professor Robert Frank and Sophie Hao for their support and guidance.

REFERENCES

Gyanendra, M. 380,000+ lyrics from MetroLyrics. Retrieved April 23, 2020 from https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics. (2018).

Genius. Lyrics Scraped from Genius. Retrieved April 23, 2020 from https://www.genius.com . Unpublished raw data. (2020).

Potash, P., Romanov, A., Rumshisky, A:. GhostWriter: Using an LSTM for Automatic Rap Lyric Generation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. (2015)

Watanabe, K., Matsubayashi, Y., Fukayama, S., Goto, M., Inui, K., Nakano, T.: A melody-conditioned lyrics language model. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1 (Long Papers), vol. 1, pp. 163–172 (2018)

Malmi, Eric, et al. "Dopelearning: A computational approach to rap lyrics generation." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016.

APPENDIX

Link to the github repository with our code: https://github.com/danielhorizon/lyrics-genreation.