Neural Language Model Song Generation

Mason Heaman **Danielle Phillips**

Abstract

In this study, the neural language model GPT-2 and bi-gram baseline are utilized as methods of song lyric generation. The neural language model is trained with the goal of generating language representing the complex nature of songs. This model takes on an even more challenging task as it is trained to replicate a certain genre of music. As expected, lyrics generated with the neural language model drastically outperform those of the baseline in terms of perplexity. In the generated samples, bigram generations obtain an average perplexity *PP* = 975.23 compared to an average *PP* = 61.27 in GPT-2 generation. The authors focus their work on a generated corpus of USA's Top Classic Rock Songs, a bi-gram baseline model, a trained GPT-2 model, and generated lyrics.

1 Introduction

Song lyrics and poetry provide a complexity that is arguably higher than other forms of conversational or informative literature. This complexity is due to the common use of metaphors, clever rhyming, and repetition. This study aims to discover the extent to which a neural language model can understand these characteristics. Using this understanding the model will attempt to generate language that falls into the song lyric format. Furthermore, the classification of songs into genres provides an interesting question for our study. Can a model be trained to not only replicate the structure and complexity of a song, but also to generate language that represents a specific genre? If so, song writers may be provided an empirical measure of the structure and ideals behind a genre of music. This may also present a method of analysis for studies investigating the societal biases supported within a given genre of music. Therefore, this study will benefit both academia and lyricists. At a high level, the approach will be broken into four key processes. Creating a corpus, establishing a baseline, training a neural language model, and lyric generation/analysis. This structure proved to be successful and should be followed with further extensions to this study.

2 Related Work

Thorough research was done in order to understand the proper use of the GPT-2 model for this project. OpenAI, the company behind the line of GPT language models, provides a GitHub page with both the GPT source code and model characteristic information used for this project. Not only did it include the needed files to download the model, but incorporated reading to learn about GPT-2. Solaiman and other authors who contributed to OpenAI's final blog post gave a brief description of their findings when working with the model after a nine-month period (Solaiman, 2020). This language model was made to serve the purpose of obtaining a better understanding of behaviors, capabilities, biases, and constraints of large-scale generative language models. The GitHub page provides a deeper understanding behind the use of the GPT-2 model, aiding in the implementation of the song generating process.

As for the python libraries utilized, the Genius API was used to retrieve the lyrics for each song, given an artist name and song title. The API was integral in the retrieval and sending of the data to a generated corpus. Inspiration for this form of lyric scraping was provided by Raizel Bernstein's article (Bernstein, 2020). Throughout the article, Bernstein provided examples and graphs to properly retrieve song lyrics.

3 Approach

The approach follows a structure that may be broken into four distinct sections. Creating a corpus, establishing a baseline, training a neural language model, and lyric generation/analysis.

To train both the baseline and neural language models, a large corpus of song lyrics must be created. This study chose to focus on a singular genre to allow the model to learn its characteristics over an effective period of time. The genre chosen was American Classic Rock (ACR). Song titles and artists were collected from a source of the 1000 most popular ACR songs (Rock n Roll). Using the collected song identifiers, the Genius API was then utilized to efficiently collect lyrics for each available song. The available song lyrics (*N*=932) were written to our corpus and separated by the text delimiter "<|end of text|>". Collection resulted in the respective line and token counts (*L*=44,571; *T*=381,700).

To establish a baseline from which to compare the state-of-the-art neural language model, a bigram model was produced from the lyric corpus. In order for bi-grams to be calculated correctly, preprocessing the text by eliminating the text delimiter "<|end of text|>" and punctuation (! , . ? etc.) was a necessity. All bigrams were collected from the dataset. The conditional probability of each bigram was then calculated. This provided a probability distribution from which our song lyrics may be generated.

4 Experiments

Using the ACR lyric corpus, language generations were made with both baseline and state-of-the-art language models. Different methods were taken for each model to receive a similar format of generation.

Four attributes were adjusted to ensure that bigram generations followed the structure of song lyrics: *stanza number, stanza length, line length,* and *top_k*. While the first three attributes control the physical structure of the generated lyrics, *top_k* will control diversity. This attribute will determine how many words will be considered when determining the next word to generate. As an example, with a *top_k*=10, the next word to be generated will be randomly selected from the top 10 most probable words that follow the previous. Attributes were randomized or set in the following manner to provide a varying structure (*SN*=2-5*;* *SL*=2-10*;* *LL*=8) while *top_k* was set at 25, 50, and 75 to find the optimal diversity in respect to perplexity. Generations (*N*=20) were then made with the noted parameters using the following method:

1. Begin sentence with a random token from corpus
2. Use bigram probability distribution to find top_k most likely tokens to follow
3. Select a random index of the most likely tokens and append to the sentence
4. Return to step 2. until target length reached

GPT-2 proved to be capable of learning the physical structure of songs, separating output into intuitive lines and stanzas. Therefore, only two attributes were experimented with to provide varying outputs from the model: *temperature* and *top_k*. Temperature controlling randomness in the Boltzmann distribution, and *top_k* controlling diversity similar to the bigram model. To briefly describe the *temperature* attribute, a lower value will result in a more confident model while a larger value will produce a smoother probability distribution (Shenk, 2017). There are pros and cons to be seen with a high or low temperature. For a low temperature, the model will be more confident on which tokens are likely to follow the previous, however, this will result in less diversity as the model is less likely to sample from unlikely candidates. Conversely, a high temperature will result in a much smoother probability distribution that will allow for a more diverse set of possible candidates, but this heightens the probability of a word being chosen that does not logically follow the previous. The *top_k* will be similarly set at 25, 50, and 75 in hopes of establishing a similar diversity to that of the baseline model. GPT-2's default *temperature*=1 was chosen with the same goal.

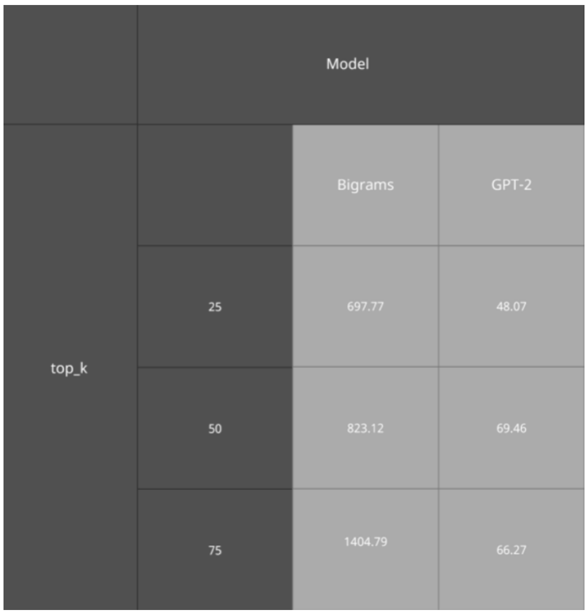
5 Results and Discussions

Results for this study will be reported under two different performance analyses: Perplexity and human labeling.

As mentioned, *top_k* values were varied on both models at values of 25, 50, and 75. This variation was done to find the *top_k* that minimized perplexity in generated samples. Perplexity is defined as the "inverse probability of the test set, normalized by the number of words". The perplexity of a sentence W can be represented with the following equation :

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_n)}}$$

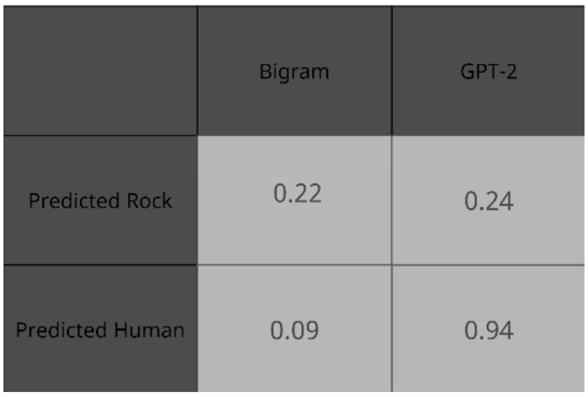
The results of the experiment are documented below, with the inner-most values being perplexity:



	Model		
top_k		Bigrams	GPT-2
	25	697.77	48.07
	50	823.12	69.46
	75	1404.79	66.27

It can be concluded that an increase in *top_k* generally results in a higher perplexity, especially in the bigram model. These results are to be expected, as a higher *top_k* results in a larger set of possible tokens to follow the previous. Therefore, there is a higher probability of choosing a token that is not a likely successor. Ideal *top_k* values for both models was found to be 25. Average perplexities calculated using all samples were *PP=975.23* and *PP*=61.27 for Bigrams and GPT-2 respectively.

Performance of the two models was also gaged with a human labeling approach. A Google Form sent out to Western Washington University students allowed participants to decide if a song was generated with by a human or AI. This will provide insight on which language model is able to better replicate human generated song lyrics. The participants were also given an open-ended question allowing them to guess the genre that the song is modeling, giving a quantitative measure of the ability for our models to represent ACR. Given the small sample size (*N*=10), this test serves only as a support towards the claims made with the more concrete measure of perplexity:



	Bigram	GPT-2
Predicted Rock	0.22	0.24
Predicted Human	0.09	0.94

The performance of the two models follows a similar pattern to perplexity, with GPT-2's probability of being label human is over 10 times greater than that of the baseline model. The models perform similarly in terms of genre modeling, however, which could point to necessary extension of the study involving fine-tuning/corpus expanding discussed in Future Work.

6 Conclusion

The neural language model GPT-2 and a bi-gram baseline model were used to generate song lyrics based off a corpus of ACR songs. GPT-2 performed markedly better than the baseline in terms of perplexity, scoring over 10 times less perplex than the bigram generations.

7 Future Work

A possible extension to this study would be to fine-tune the GPT-2 model. Fine-tuning consists of making small adjustments to the model to improve the results. With the GPT-2 model, there are numerous ways to fine-tune it as there are methods available in the codebase. OpenAI released an article where they fine-tuned the GPT-2 model based on human preferences by asking human labelers for the best and accurate result. This would be great to consider for the project when making the comparison between the GPT-2 model and bi-gram baseline.

Corpus creation may be extended to collect more songs and support the modeling of multiple different genres. A possible means of collecting song identifiers would be to use the Spotipy python library in conjunction with the Spotify API.

A final extension to this study may involve the use of neural language models to generate titles for the generated songs. This will further test the models to assess if they are able to capture the main idea and create an all-encompassing title for the lyrics.

References

Bernstein, R. (2020, November 28). How To Access and Use the Spotify and Lyric Genius API. Retrieved from https://raizelb.medium.com/how-to-access-and-use-the-spotify-and-lyricgenius-api-a5563ef7f7b1

Jurafsky, D., & Martin, J. H. (2014). *Speech and language processing*. Harlow: Pearson Prentice Hall.

Shenk, Justin What is Temperature in LSTM (and neural networks generally)? Retrieved from https://cs.stackexchange.com/questions/79241/what-is-temperature-in-lstm-and-neural-networks-generally

Openai. (2019, November 26). Openai/gpt-2. Retrieved from https://github.com/openai/gpt-2/blob/master/model_card.md

Rock n Roll America's Top 1,000 Classic Rock Songs (Our Base Song List). (n.d.). Retrieved from https://www.rocknrollamerica.net/Top1000.html

Solaiman, I. (2020, September 04). GPT-2: 1.5B Release. Retrieved from https://www.openai.com/blog/gpt-2-1-5b-release/

Ziegler, D. (2020, September 02). Fine-Tuning GPT-2 from Human Preferences. Retrieved from https://openai.com/blog/fine-tuning-gpt-2/