Computational Hip-Hop Lyrics Generation and Transformers

Essam Babu (essam.babu@mail.utoronto.ca)

University of Toronto

Abstract

Hip Hop lyrics contain many natural language processing challenges that encompass many aspects of human written abilities. The objective of this paper is to model the process of generating hip hop lyrics using machine learning networks. A large-scale Transformer model GPT-2 was fine-tuned on a large database of hip-hop lyrics. The model was then evaluated based on rhyme density and unique word count in their generated lyrics. The model was shown to somewhat model the structure of hip-hop lyrics but not better than any human comparison.

Keywords: Natural Language Generation, Transformers, Transfer Learning, Lyrics**,** Natural language processing

Introduction

The ability to make sense of our world through different forms of storytelling is seen universally across cultures. It is a distinctly human trait to tell stories and share our experiences in as many creative ways as possible. Songwriting is writing ability that requires creativity such that you can develop a narrative or image while being constrained by a musical format of some kind. The genre of Hip-hop music provides several constraints on the songwriting process that make this a challenging style of music to write well. Hip-hop song lyrics have the objective of trying to form rhymes with their words while developing a narrative throughout a given song. Hip-hop music lyrics are also typically more verbose in that the unique words per line and words per minute are on average higher than other genres of music (Mayer et al, 2008, p. 341). This makes generating hip-hop lyrics an especially interesting form of natural language generation with a measurable objective of rhyming, a wealth of text data, and important natural language challenges such as understanding expressions, idioms, and slang-vocabulary. The objective of this study is to computationally model the ability to generate song lyrics, specifically hip-hop lyrics.

Natural Language Processing (NLP) is a branch of artificial intelligence that has made great advancements recently due to the development of a novel encoder-decoder style neural network called a Transformer (Vaswani et al, 2017). Previously, recurrency and convolution based sequential neural networks were standard for NLP tasks, but their sequential nature made it hard to capture long-term dependencies between tokens that could be relevant to each other but are far apart in the word vector. The attention mechanism (Bahdanau et al. 2014) was introduced into RNN and CNN neural networks to capture relevant long-term connections between all tokens in an input sequence or other representations such as in the case of images. The attention The attention mechanism creates dependencies between all tokens within an input sequence that then allows you to hone in on the most relevant information. Since this was implemented in sequential neural networks, the attention process needed to occur at every step of the input, making this computationally infeasible. This computational worry was addressed in Google's *Attention is all you Need* (Vaswani et al, 2017) where the Transformer network was introduced, which showed a novel neural network architecture that could compute the attention mechanism in parallel for every input token at once while foregoing recurrency entirely.

The transformer was originally introduced to handle a common category of NLP tasks: the sequence-to-sequence (seq2seq) problem, where you convert a sequence in one domain, such as an English sentence, to another domain, like the same sentence but translated into French. Beyond machine translation, you could use a seq2seq approach in other domains such as question-and-answer domains, or other domains involved in generating text conditioned from another piece of text (Chollet, 2017). Transformers implement encoders and decoder networks comprised of self-attention modules which perform the attention calculation in parallel, followed by a feed-forward neural network.

Since the original transformer, many large-scale transformer models have come out into the public to achieve state of the art performance on NLP tasks across the board. Examples include Google's *Bidirectional Encoder Representations from Transformers* or *BERT (*Devlin et al. 2018*)* and OpenAI's Generative Pre-trained Transformer or *GPT* Model (Radford et al., 2019; Brown et al. 2020). These large-scale transformer models are trained on large corpus' of text where they learn to capture an inner representation of an entire language. This language model can then be further trained on a narrow task domain using a refined set of data to then achieve superior results on downstream tasks with less data than if you were to train directly on the downstream task in the first place. This technique known as transfer learning allows users of these large scale pre-trained models to achieve significant performance with little time and data (Raffel et al., 2019). This technique is significant important because most language tasks do not have a lot of

data points to cover its entire hypothesis space, making over and underfitting to these problems quite common.

In this paper, we will implement GPT-2 and fine tune the model on the task of generating hip hop lyrics. Computational hip hop lyrics generation is a NLP task that has been explored before using RNNs and Long Short-term memory networks (Malmi et al. 2016, Potash et al. 2015) and more recently with Transformers (Nikolov et al. 2020)

Materials and Methods

To see the full implementation and code repository https://github.com/EssamB13/gpt2-ghostwriter

Data

The dataset was created by scraping lyrics from Genius.com using a Python API called LyricsGenius. There was a total of 5601 songs scraped from 141 hip hop artists corresponding to roughly 40 songs from each artist. The set of artists was initially sourced from a Wikipedia page listing many hip-hop artists and alternative hip-hop artists[2] which contained a wide variety of artists from different eras and sub-genres of hip hop.

The pre-processing of the lyrics was kept to a minimal. The lyrics were collected into one large text file where each artist's collection of song lyrics would come one after another, separated by a start and end of text token which looked as followed:

```
<|startoftext|>

LYRICS OF ONE SONG

<|endoftext|>|
```

Figure 1 Example of pre-processing

There was also another set of lyrics gathered that kept the headings of the sections of music which were provided by Genius's transcription of the lyrics:

```
<|startoftext|>
<< ARTIST – SONG_TITLE >>
[Verse 1]
LYRICS OF VERSE

[Chorus]
LYRICS OF CHORUS
…
<|endoftext|>|
```

Figure 2 Example of supervised preprocessing

The lyrics had these tokens added using the *HuggingFace Transformers* Library, which provides a Tokenizer that prepares the data in the specific format needed for GPT-2.

The data was then split into two groups: Training and validation, where 4481 songs, or 80% of the dataset, was used for training, and 1120 songs were used for validation during testing. The training set was prepared such that it would be sampled from randomly, while the test set was looked at sequentially since this is not significant for validation.

Model

The model's training was developed in Python using PyTorch, and the model itself was supplied by *HuggingFace's Transformers* library. *Transformers* is an open-source repository of NLP models and datasets that contains many versions of popular large-scale transformer models along side the scripts and tools needed to use them.

The model used was *Transformers' GPT-2 (Small)* version, which contains 117 Million parameters and takes an input sequence of length 768 words. GPT-2 (Radford et al., 2019) is a large-scale Transformers based generative model, which contains only the decoder portion of the original Transformer encoder-decoder architecture. This model is was trained to predict the next word in a sentence, and was trained on a large corpus of internet data such as Wikipedia articles and Reddit.

Training

The GPT-2 model was trained on a single NVIDIA GeForce RTX 2070 Super, which is a GPU that can leverage CUDA cores to train deep learning models faster. Hyperparameter choice was mostly found by experimentation, and no hyperparameter tuning was utilized. The model trained for 3 epochs with a batch size of 1 since a single piece of text may contain well over 700 tokens, this made it hard to increase batch size with only one GPU. Gradient accumulation may be a way to overcome this constraint. Learning rate scheduler was handled by the AdamW optimizer (Gugger & Howard, 2018), which is an improvement from typical stochastic gradient optimization.

Two separate models were trained: one on the text data with our artist and song information, and one with artist and song information. This was done to create two different versions of a lyric's generator. The version trained on no labels, besides start and end token designations, simply learned only how to predict the next work based the words it saw before. This model's goal is to simply rap with no reference to whose style they could be imitating: this model is called the *Freestyler*. The other model trained on the

labeled song lyrics data is then able to associate the lyrics with the artist's name themselves, and thus we can then prompt this model to imitate another artist's style, known as *Ghostwriting*. The *Ghostwriter* model is also trained on knowing what exactly a chorus is and what is a verse, etc. The *Freestyle* model is fully capable of learning of what distinguishes a verse from a chorus since they are often different enough stylistically to tell apart by text alone, but it would never be explicitly told this information.

Lyric Generation

Once the model was trained adequately enough, the models can then be used to generate lyrics if conditioned on a prompt. The prompt could be as simple as:

<|startoftext|>

Figure 4 Example of Unconditioned Prompt

which is enough to begin predicting the next work, which in this case, would be predicting the first word of a song. It could be as complicated as:

<|startoftext|>
[Verse 1]
Already woke, spared a joke, barely spoke, rarely smoke
Stared at folks when properly provoked, mirror broke

Figure 5 Example of more elaborate prompt

Which would give the model more priors to better inform the next work prediction. The *Ghostwriter* model can also be used to imitate a specific artist, let the artist's name be *Big Drama*, with the following prompt:

<|startoftext|>
<< Big Drama -

Figure 6 Example of Prompting a specific artist's style

Which would prompt the *Ghostwriter* model to then generate a title for this *Big Drama* song and then generate lyrics.

To generate the song words, a sampling technique has been employed to sample from the model. *Top-K* sampling (Fan et al., 2018) is a simple sampling scheme where the *K* most likely next works are filtered and the probability mass is distributed among only those words (von Platen, 2020).

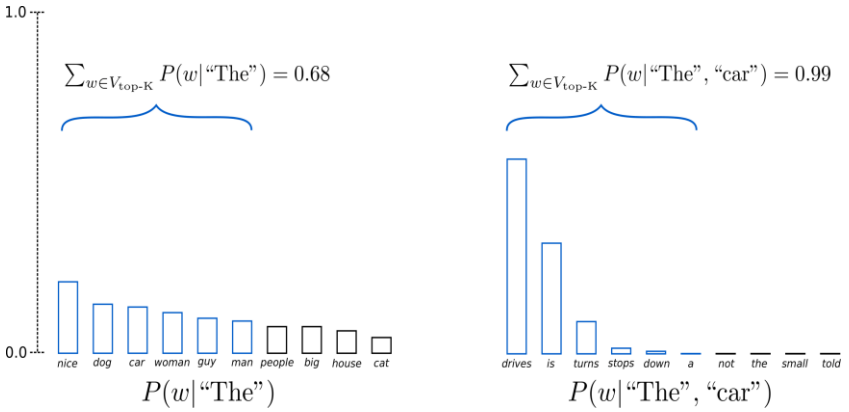

Figure 3 K= 6 Example of Top-K sampling Redistribution

Another sampling technique used to generate words is *Top-P* or *Nucleus sampling* (Holtzman et al. 2019). Instead of sampling only from the most probably K words, Top-P sampling creates the smallest possible set of works whose cumulative probability exceeds the probability p. The probability mass is then redistributed among this set of words. This allows for the word set to be dynamically increasing and decreasing according to the next word's probability.

The combination of both sampling techniques, specially with n=0.92 and K=75 produced the most "human" like lyric generation, and these values were found through experimentation.

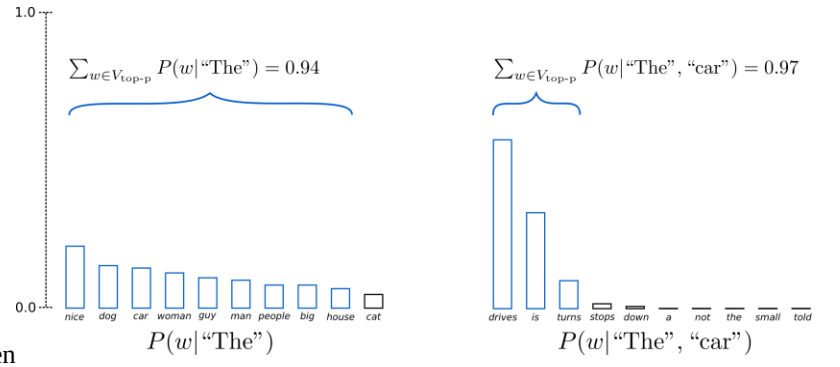

Figure 7 p = 0.92 example of Top-P Sampling

Evaluation

To evaluate the lyrics generated, there were two main metrics used: Rhyme density and unique word count. Rhyme density is a metric developed by Malmi et al. (2016) which is a measure of the average length of multi-syllabic assonance rhymes, or rhymes where the vowel sounds match despite different consonant sounds surrounding the vowel. An example of a multiple syllabic assonance rhyme:

> **My heart and tongue are fighting, my mind is undecided**
> **It's not like Trump and Biden it's more like something private**

Where the colors represent different vowel sounds. To measure Rhyme density, the procedure is as followed:

1. Run the lyrics transcript through g2p (eSpeak) to get a phonetic version. Remove non-vowels.
2. For each word:
   Find the longest matching vowel sequence near the word. (This is a multisyllabic assonance rhyme.)
3. Compute the density by averaging the lengths of the longest matching sequence over all words.

To perform this task, I utilized Malmi et al. (2016) open source Python script which calculates this called *Raplyzer* (which has been included in the repository). For reference, a high rhyme density across all an artists songs was seen by Inspectah Deck at 1.187, where as a low score of 0.87 was obtained by comedy musical group The Lonely Island.

I have used unique word count as an indicator to see the range and diversity of speech in the models. This is a statistic previously explored in hip hop by Daniel (2019).

To assess the *Ghostwriter* model, 4 artists were randomly chosen from the artist set alongside a popular song of theirs. The first four lines of each song were used as a prompt. The model then generated 50 examples of a full set of lyrics prompted from each song's beginning, and the best five examples were hand selected. Then the average rhyme density

To assess the *Freestyle* model, 5 artists were randomly chosen alongside their 5 most popular songs, where each artist had their average rhyme density and average unique word per song calculated and compared to 5 songs the *Freestyle* model generated with the generic start of text prompt, hang chosen from a total set of 50 songs that the Freestyle model generated.

Results

The following results were obtained from the Ghostwriter Rhyme density measurements:

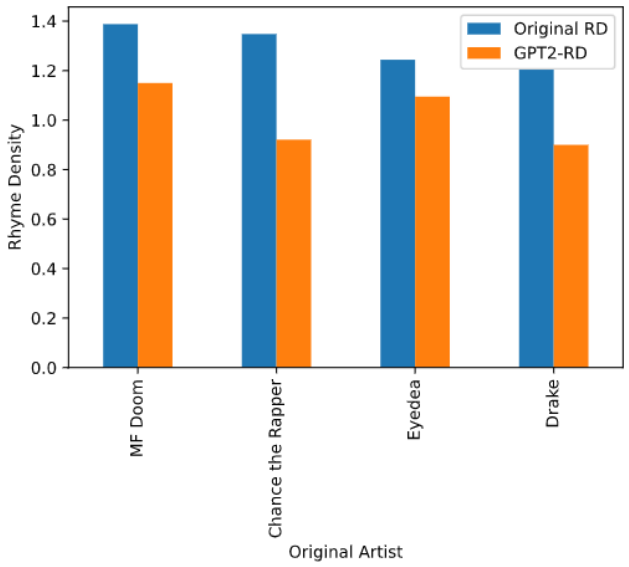

Figure 9 Ghostwrite Rhyme Density Performance. Orange bar measures the Ghostwriter performance in using that artists song as a prompt vs. the actual RD of the original song.

And the following was obtained from the Freestyler Rhyme density measurements:

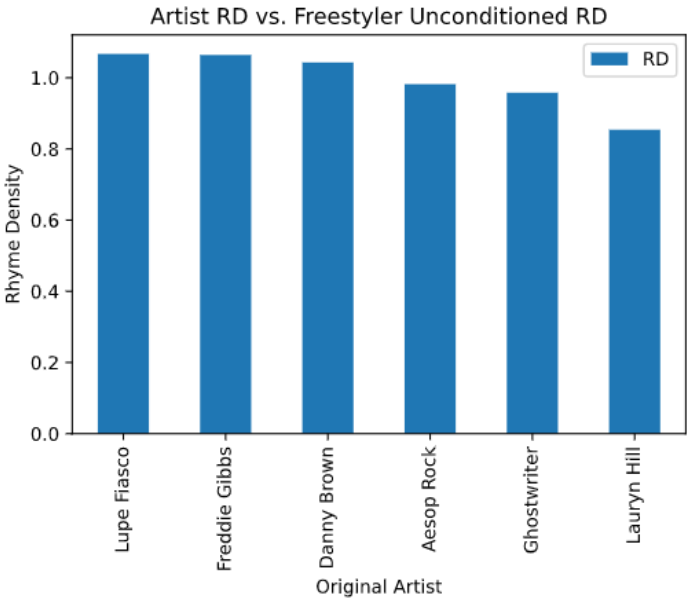

Figure 8 Freestyler Rhyme Density Performance vs. Artist Avg. RD.

The same process was completed for the *Ghostwriter's* assessment of the unique word count, where the best 5 examples prompted from each song had their unique word count averaged and compared to the original song it was prompted from:

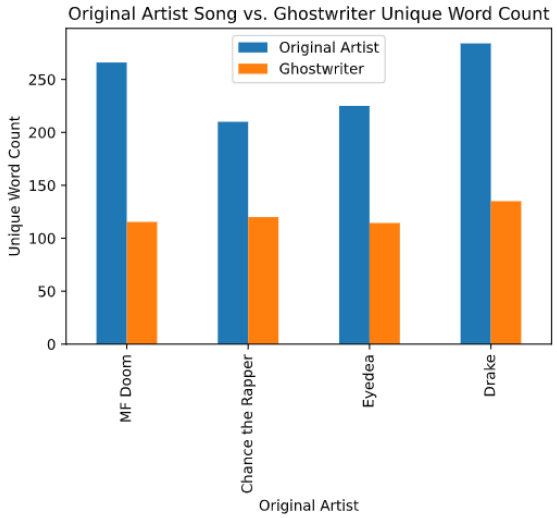

Figure 10 Ghostwriter's Unique word count vs the original song's unique word count

The freestyler's average unique word count was calculated for its 5 generated songs was then compared to the other chosen artists's average unique word count:

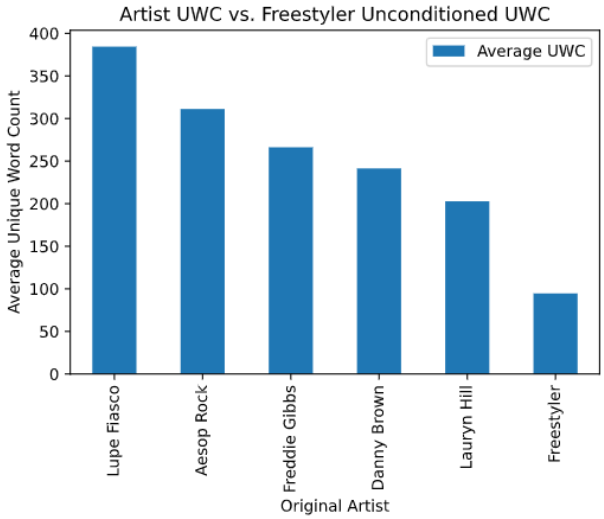

Figure 11 Freestyler Average Unique Word count vs. Artists

Finally, I will display an example of generated lyrics from the Ghostwriter and the Freestyler:

I'm in love with you
You say you a baller, I say you a hater
You say you can ball in the slammer
And I know you ain't gon' do that if you ain't gon' ball today
You say you want a baller, I say you a player
You say you a baller, I say you a player
And that's all I know 'cause it's you
I know you ain't gon' do that if you ain't gon' ball today
And you say you want a baller, I say you a player
You say you a baller, I say you a player
And that's all I know 'cause it's you
I used to be taught by a young boy who was dumb as hell
In high school I was an early fool

Figure 12 Freestyler Example Lyrics, p = 0.92, k=50

[Verse 1]
Uh, you underestimated greatly
Most number ones ever, how long did it really take me?
The part I love most is they need me more than they hate me
So they never take shots, I got everybody on safety
Hollow-tips stay on my body
Be careful who you askin' about
D-town's only folk outside the famous functions
Still walk around pissin' off history
Bodies lie down for service
Grown folks in suits and gowns
You can't fake good, like the old Adidas
We get high as a hell of a drug
Everybody's a victim of a fist fight
Everybody's a victim of a drug
Everybody's a victim of a fist fight

Figure 13 Ghostwriter conditioned on the first four lines of Drake's *5AM IN Toronto, Bolded lines are the prompt. p=0.92, k=50*

Discussion

The results show that while the model can generate lyrics that are song like in nature, their performance is noticeably worse than human lyrics. The Ghostwriter RD performance was able to closely align with the real human performance, showing that the Ghostwriter model is somewhat capturing the rhyme technicality of the artist it is Ghostwriting for. Similarly, the Freestyler was able to outperform one human lyrist in Rhyme density.

The unique word count performance is where you see the main flaw in these models in that they tend to end up in repetitive sentences that often copy each other. This may be since the model was learning repetitive lyrics from the chorus' it was scanning, and perhaps future versions of this model could simply remove chorus' altogether to focus on technical rhyming lyrics that occur in the verse only. The performance of both models, alongside their repetitive sentence structure mean that the models are overfitting to this dataset and would likely need more data to see more robust results.

While no official measurement was taken, the semantic coherence of the lyrics output is quite low, and while sometimes the lyrics can flow well together, the lyrics are very noticeably not very intelligence and coherent at all. Syntactically, the sentences are mostly fine. Perhaps a bigger model, such as GPT-3, trained on a bigger dataset could provide more convincing results.

Conclusion

In this study we have explored natural language generation using large scale pre-trained Transformers in the space of hip-hop lyrics generation with the objective of trying to rhyme. While the model has learned the syntax of rap music, it has not learned how to rhyme well or maintain semantic coherence.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*

Chollet, F. (2017). A ten-minute introduction to sequence-to-sequence learning in Keras. *Retrieved April 15, 2021, from https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html*

Daniels, M. (2019, January 21). Rappers, sorted by the size of their vocabulary. Retrieved April 15, 2021, from https://pudding.cool/projects/vocabulary/index.html

Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Gugger, S., & Howard, J. (2018, July 02). AdamW and SUPER-CONVERGENCE is now the fastest way to train neural nets. Retrieved April 15, 2021, from https://www.fast.ai/2018/07/02/adam-weight-decay/

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751.

Malmi, E., Takala, P., Toivonen, H., Raiko, T., & Gionis, A. (2016, August). Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 195-204).

Nikolov, N. I., Malmi, E., Northcutt, C. G., & Parisi, L. (2020). Conditional rap lyrics generation with denoising autoencoders. *arXiv preprint arXiv:2004.03965*

Potash, P., Romanov, A., & Rumshisky, A. (2015). Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1919-1924)

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Rudolf Mayer, Robert Neumayer, and Andreas Rauber. (2008). Rhyme and style features for musical genre classification by song lyrics. *In Proceedings of the 9th International Conference on Music Information Retrieval, pages 337–342.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*