Frank Costantino

CS 499 - 001

May 9, 2022

Hip-Hop/Rap Lyric Generation Using GPT-2

Contents

Abstract

Rap generation, which aims to produce lyrics with stylistic rhyme patterns and rhyme schemes, requires the extensive modeling of Rap and Hip-Hop lyrics. Similar to Potash et al. 2015, the goal of this model was to generate lyrics that are similar in style to a given rapper, but not entirely identical to existing lyrics. This model makes use of existing explicit templates for lyric generation that are given through the retrieval of song lyrics used in training time. This report explains the process behind rap lyric-text generation using GPT-2.

GPT-2 in itself stands for Generative Pre-trained Transformer 2. Generally, GPT-2 is a very large transformer-based autoregressive language model with over 1 billion parameters that is trained on a dataset of 8 million web pages. Transformers utilize attention which relates different positions of sequence tokens (e.g. words in a text sequence) to compute a representation of the sequence. This idea of attention and more specifically self-attention has been shown to perform well on a variety of language modeling tasks and it has become a foundational part of complex sequence modeling and dependency modeling (Vaswani et al. 2017). Although GPT's use cases are plentiful, its deep neural architecture allows it to excel at text-generation. In this project, GPT-2's main usage comes down to successfully predicting next words to create coherent rap lyrics. Unsupervised learning takes place as the model trains on a user-selected rap artist, with the model being later fine-tuned. Since the training data is relatively small, this process nets the largest possible net gain for this text-generation task.

Implementation: Data Preparation

This proposed implementation discussed in this report uses GPT-2-Simple, which is a Python package that wraps the existing scripts for GPT-2 fine-tuning and text generation provided by the OpenAI team (Woolf 2022). Specifically, this implementation utilizes the small 124M - a reference to the number of the model's parameters - with a size of 500MB. All model training, text generation and model evaluation is done using Google's Colaboratory notebook, which provides a powerful GPU for fine tuning. Data aggregation and data cleaning for model training was completed using Python scripts.

Data preparation was performed using *LyricsGenius*, which wraps usage of the Genius API for gathering lyrics data stored on Genius.com. Initial data collection efforts were done manually by simply compiling a text file of 76 artist names separated by newlines that the implementor personally listens to or has heard of in `.\data\list_rappers.txt`. This file serves as the basis for data collection and the list from which users can select an artist's name from. Implementation of data aggregation is sourced from the code in the files `fetch_lyrics.py` and `rap_model.py` in the project submission. Using the `Lyric_Grabber` class of `fetch_lyrics.py`, which encapsulates a single user-selected artist and serves to fetch any desired number of songs for an artist up to 100. Artist names that are not spelled correctly, including accented characters and typos, are corrected to the best of the ability of the *LyricsGenius* package, which handles all cases faced in the implementation. Each artist has a single text file with exactly 20 songs, where each song entry includes special beginning-of and end-of song tokens to denote the beginning and end of songs ("<|BeginSong|>" & "<|EndSong|>"). Next, separated by a newline follows the song's title and any tags that denote the song's introduction, choruses, verses, or bridges. A stipulation here is that bridges will not appear at the beginning of songs only within the middle

of songs. Introduction tags list all artists featured in the song while verse tags only list the artist associated with that particular verse. Like introduction tags, chorus and bridge tags list all artists that sing within said chorus or bridge. For songs that feature only a single artist, the artist's name is not listed at all, and would only be known from the name of the text file otherwise. Chorus, verse and bridge tags appear as they happen within the lyrics. When the song's lyrics conclude, not including any outro musical accompaniment, the "<|EndSong|>" is inserted with two newlines and the lyric text file is closed. This process is repeated for all 76 artists in `.\data\list_rappers.txt`, which brings the song total to 1520, although only 20 songs are used at any time due to the implementation guidelines.

Implementation: Model Fine Tuning

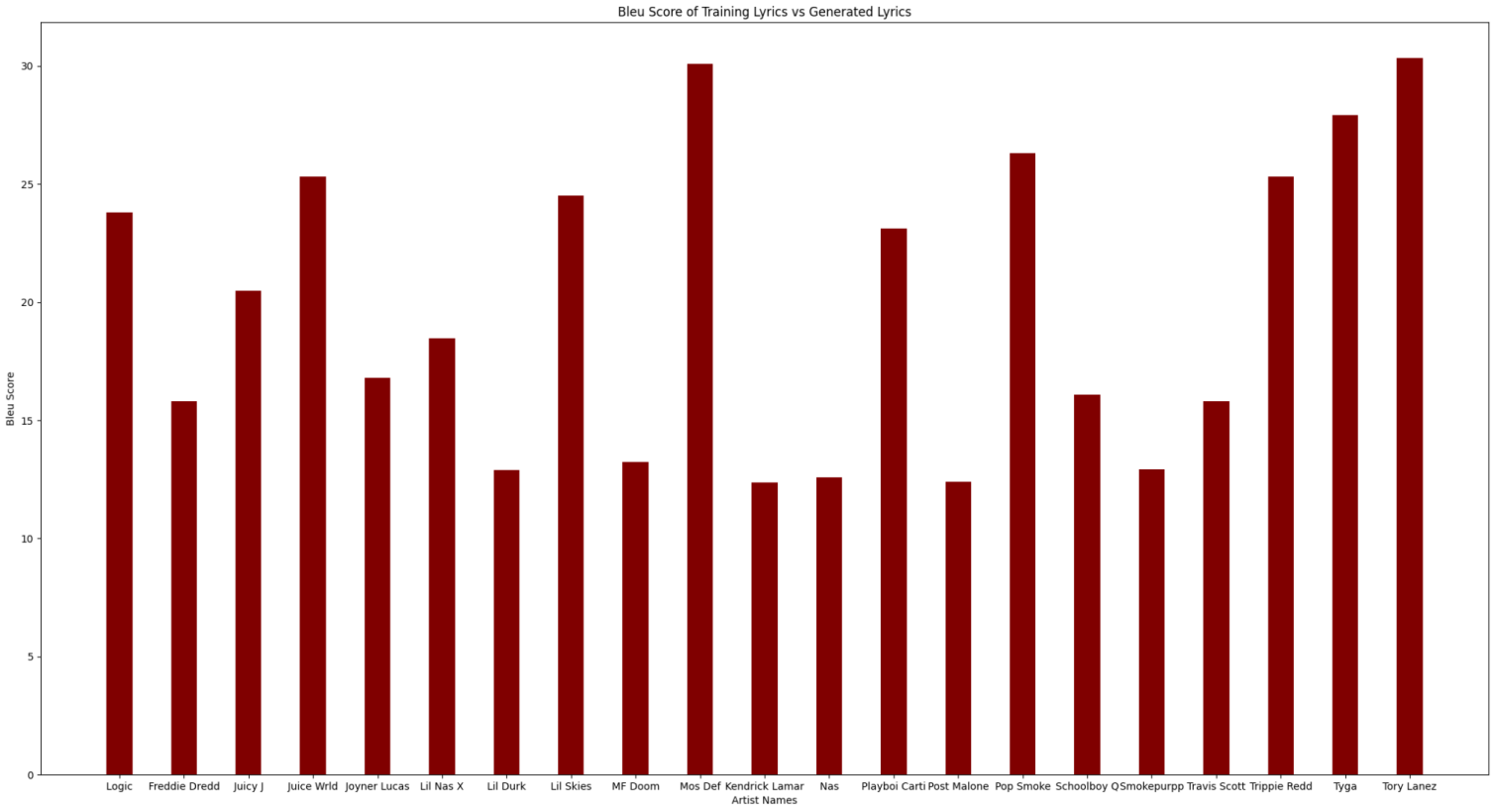
Model fine tuning takes place within both the `rap_model.py` and Google Colab notebook. Logically, the implementation flows from the former to the latter. Beginning in `rap_model.py`, the Lyric_Grabber class is used to write all artist lyric files to a single directory entitled `lyrics`. Users are then prompted for the desired artist he or she wishes to generate lyrics in the spirit of. If the artist is not found, the user is prompted repeatedly. The implementation then moves to the Google Colab notebook for actual model training, text generation and evaluation. Within the Google Colab notebook entitled `cs499-final-project`, all lyric files are copied from the private Github repository with the same name as the Colab file, which contains all lyric files as they are downloaded. The implementer's Google Drive is mounted, and GPT-2 small 500MB 124M model is downloaded directly to the drive. With all lyric files mounted as well, a persistent TensorFlow session is created, which stores the training configuration for the Colab notebook. In the case of hyperparameter tuning, model checkpoints are saved every 500 steps, with 1000 steps total. Checkpoints are saved and reused during subsequent training

iterations, hence overall training steps are low to save time during training. Other hyperparameters specified are learning rate at 0.0001, which is standard for this model size and output samples are generated every 200 steps. Training loss across multiple training runs ranges approximately from 0.04 to 3.49, which will be discussed further in the next report section on evaluation.

Text samples are then generated of length 250 tokens. This token size helps alleviate blank lines accompanied by artist names. Model temperature is set to 0.7, which keeps the text coherent and sensical, versus outlandish. All output sequences are truncated with '<|EndSong|>' tokens. Generated text samples are limited to top-k guesses with a default k of 0 at the time of submission. Nucleus sampling is used to limit the generated text samples to a cumulative probability of 0.9, which was determined by trial and error. A batch size of 1 is used to alleviate incredibly long generation times.

Results & Analysis

Objective evaluations were made using Bleu score. Bleu score evaluations are made between the generated text written to a file and the artist lyric file used during training time. Lengths for each of these files needed to be truncated to the size of the generated text file so that the Bleu score evaluation could be made. Subjective evaluations that accompanied these measurements included a thorough reading of generated lyrics to see if the text verbatim copied lyrics used in training time and for rhyme consistency and rhyme frequency. Bleu scores between the training data and the generated text are seen in the figure below for a number of artists post-training lyric generation:



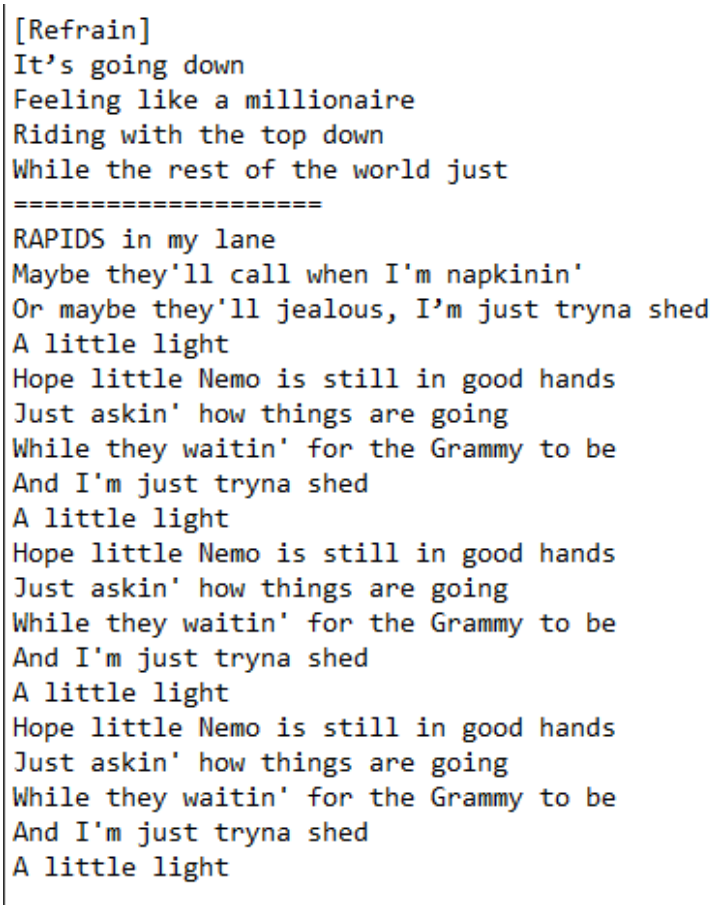
As seen by the figure above, it can be noted generally that the Bleu score calculated between the training lyric data and the generated lyric text ranges generally from about 11 to 30. Lyrics within this range are generally easily interpretable and from objective hand-evaluations

done by perusing generated lyric files, this fact can be easily corroborated. Some text samples generated for evaluation are included below in Figure 1.

[Figure 1]

```
RAPPLES
My name is Logic, if you don't know by now, I'm always on my grind
And at this moment in time, I’m on a road when I write this rhyme
Sitting behind Raheem Devaughn while he’s passed out
'Bout to hit the station 'fore our gas is out
Look outside the window, I see shorties with they asses out, oh my
Good god, you know we fly
Day dreaming out the window, watch how buildings pass me by
See I ain’t signed but at this point in time
I'm feeling like the 3-6-8th
Wonder of the world, just might steal your girl
All I do is rhyme and get money, that’s my repertoire
Holla at honeys, "Voulez-vous coucher avec moi, ce soir?"
Baby what’s the deal, have you ever heard of some shit so real?
(Cudi what up?)
```

```
[Chorus: Logic]
I can feel you inside me
Feel your lungs on the inside
Possess the strength to fight
I'm on the outside, but inside I'm burning
Feeling like I'm outsmokin'
In my head, I'm saying, "Fuck 'em all"
Can you imagine how many enemies I'mwacking with you
Just so you fuckin' people know?
[Outro: Thalia]
My name is Artanis, and I'm the frontman for my label Thalia
I'm an A-list artist when you lookin' for something
I'ma play the keys and then sing the blues
My raps are the clean lines, and my raps are the raps
No, I'm not an A-list A-member
I'm an A-member
[Verse 4: Logic]
I'm the one, bitch, I'm the one
It's an intense relationship, the game-changer is over, bitch, what up, A-list A-member?
[Chorus: Logic]
Hold up
Been on the low, been on the high, been looking for something
And I'm back again, and this time it's R&B
Livin' my life while you young and woozy
All alone and alone in my own zone while others treat me the same
All alone and alone in my own zone while others treat me the same
[Chorus: Logic]
Hold on, breathe, breathe, breathe, breathe, breathe, breathe
Hold on, breathe, breathe, breathe, breathe, breathe, breathe; Breathe
```



```
[Refrain]
It's going down
Feeling like a millionaire
Riding with the top down
While the rest of the world just
====================
RAPIDS in my lane
Maybe they'll call when I'm napkinin'
Or maybe they'll jealous, I'm just tryna shed
A little light
Hope little Nemo is still in good hands
Just askin' how things are going
While they waitin' for the Grammy to be
And I'm just tryna shed
A little light
Hope little Nemo is still in good hands
Just askin' how things are going
While they waitin' for the Grammy to be
And I'm just tryna shed
A little light
Hope little Nemo is still in good hands
Just askin' how things are going
While they waitin' for the Grammy to be
And I'm just tryna shed
A little light
```

[Figure 1]

Figure 1 displays lyrics generated in the style of the American rapper Logic. The first image of Figure 1 is more than intelligible, and has some rather complex rhyme patterns, minimal to no overt text errors, and has a great rhyme fluency and density. A major discrepancy in this image is the fact that approximately 90% of the lyrics generated are verbatim from the song "All I Do" by Logic. In some extreme circumstances, the model generates lyrics that are carbon copies of existing songs, or generates entire song verses verbatim. The second and third images of Figure 1 display something that is far typical of the generation, with lyrics ending abruptly and lyrics being repeated. Additionally, lyrics have no general flow, either logically or lyrically and seldom incorporate lyrics verbatim from Logic songs. Although the generally high

Bleu scores indicate that lyrics are human intelligible, this metric does not account for patterns of copied lyrics the model generates. Hand-evaluations of lyrics net much better metrics for how well the model is performing in its ghost-writing-esque task. In essence, for the sake of time, human evaluators would need to be extremely familiar with rappers the model is trained on so they can easily spot lyrics that are copied verbatim from existing artist songs.

Interestingly, much of the song's structure remains intact and is executed well enough by objective standards. Structure includes tags that indicate transitional song pieces such as choruses, introductions, outros, verses, refrains, and bridges. These tags help conduct the song's rhythm and flow with traditional beat accompaniment, but serve as markers that the model has learnt to harp on as tags in the above Figure 1 are either placed haphazardly and frequently or none at all. Regardless, a main pain point remains to be an abundant amount of overfitting. This is the result of the implementation's computational limits: the model is trained on a relatively small step size of 1000 steps for a single batch as GPU usage is limited on Google Colab and training data was limited to avoid incredibly long training times that would cause GPU usage timeouts on Google Colab. Hence, aforementioned model losses ranged between 0.04 and 3.49, where training would start with an average loss of 3.5 and end with an average loss relatively close to 0. Through these metrics for evaluation, one can conclude that the training data was not plentiful enough, as having only 20 songs per training session with approximately 25,000 tokens on average per artist text file netted less satisfactory results. Hence, the model did not effectively ghost-write lyrics, but instead wrote exact lyrics from previously existing songs.

Future Implementation Considerations

Many improvements can be made to this implementation. The computational limits of the implementation can also be seen through the usage of GPT-2 124M model, opposed to the 355M medium sized model or larger models with close to a billion parameters. Google Colab's GPU usage is limited upon repeated use, which limits the number of training sessions possible over the span of multiple days. Without the limit of GPU usage, there would be a monumental reduction in model training time, allowing for better hyperparameter adjustments such as larger step sizes, larger batch sizes and lower learning rates. Although minimal improvements can be made in the attainment of more computing power given the current implementation environment, the addition of up to more training data would undoubtedly help with the novelty of generated lyrics. The GPT-2 model used is pre-trained, but given the data collection methods used it was difficult to fully utilize the full potential of pre-training. It could be suggested to have as much as four times the amount of songs per artist for the model to train with. Data collection remains strong in original implementation but data cleaning could be improved in a number of ways. For example, the inclusion of additional special tokens beyond those indicating the beginning and ends of songs, but also special tokens to denote the end of choruses, verses, and other song structure tags. This would allow for more structured text generation that does not rely so heavily on the model to produce viable song structure.

Model evaluation can be improved as well. Model evaluation now serves only to measure the coherence of text, not any objective metrics of what is to be considered well-written rap lyrics. This was mainly done in the interest of time but also to provide a baseline metric for model performance due to implementation limitations discussed. Most evaluation metrics were subjective but ad-hoc. Subjective evaluations include evaluations of rhyme patterns and lyrical

flow - if the lyrics appear that they could be from a human-written rap song. Due to the inherent slow pace of human subjective evaluations, many computational and objective evaluation methods have yet to be implemented. Other evaluation metrics can be implemented such as perplexity scores, evaluation of rhyme density and frequency, and n-gram evaluations that measure the number of consecutive sentences with the same rhymes or words to combat repeated lyrics (Xue et al. 2021). In particular, perplexity calculations were difficult to ascertain as the GPT-2 pre-trained model's vocabulary was not available. Conclusively, major improvements can be made to data collection and model evaluation to provide a stronger basis for text generation.

Ethics

The implementation discussed is by no means a novel language model for rap generation. However, this implementation also aims to show that music lyric generation is increasingly accessible as this framework is also beneficial for generating other music genres. Due to the nature of data collection and how text is generated, the training dataset may have artist biases to certain vernacular or topics, which brings some potential risks to generated text. Additionally, the training data contains copyrighted text, and considerations to musical labels and the Digital Millennium Copyright Act (DMCA) of 1998 would need to be considered on the legal implications of the generation of novel lyrics from pre-existing copyrighted material. Questions that yet to be answered include possibilities of the outputted text being subject to copyright due to the nature of copyrighted inputted text, if generated text can be considered novel at all under the law, or other ethical implications of generating lyrics in the spirit of the artist without any affiliation or association with said artist.

Conclusion

This implementation proposes the usage of GPT-2 to generate rap songs conditioned on a small number of songs from a user-selected artist. Hence, lyrics are generated in the style of a certain artist, drawing parallels to the idea of "ghost-writing". It has been shown that this method is capable of generating coherent text with subject matter similar to that of contemporary rap songs. This approach could be modified to better explore alternative expressions of ideas discussed in rap songs. Generally, conditional text generation is applicable to a large variety of creative texts in other domains of text such as short stories, poetry, and fanfiction. Future work could explore further improving this implementation as discussed previously. Additionally, generating coherent lyrics could be looked upon not as a generation task, but rather modeled as a text-editing task of existing lyrics to create novel ideas and representations of existing rap lyrics. Hence, anyone is encouraged to apply other techniques to similar problems that this framework needs work to address.

References

Potash, P., Romanov, A., & Rumshisky, A. (2015). GhostWriter: Using an LSTM for Automatic Rap Lyric Generation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1919–1924. https://doi.org/10.18653/v1/D15-1221

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv:1706.03762 [Cs]*. http://arxiv.org/abs/1706.03762

Woolf, M. (2022). *Gpt-2-simple* [Python]. https://github.com/minimaxir/gpt-2-simple (Original work published 2019)

LyricsGenius: A Python client for the Genius.com API — lyricsgenius documentation. (n.d.). Retrieved May 8, 2022, from https://lyricsgenius.readthedocs.io/en/master/

Xue, L., Song, K., Wu, D., Tan, X., Zhang, N. L., Qin, T., Zhang, W.-Q., & Liu, T.-Y. (2021). DeepRapper: Neural Rap Generation with Rhyme and Rhythm Modeling. *ArXiv:2107.01875 [Cs, Eess]*. http://arxiv.org/abs/2107.01875