Team: Frank Costantino (just me)

CS 499 - 001

Project Proposal

March 26, 2022

Rap/Hip-Hop Controlled Text Generation

The concept of text and music generation has always been fascinating to me. Actually, last semester, my roommate and I worked on a project together that aimed to use Variational Autoencoders (VAEs) in novel music generation for Lo-Fi beats. If you’re unfamiliar with Lo-Fi, you may have seen images of the “Lo-Fi Girl” around the internet as she perpetually studies at her desk. In the same way music helps you visualize a story, lyrics quite literally tell the story. That’s where the idea of Rap/Hip-Hop text generation comes into play. This idea is the successor of last semester’s project, which will focus on an equally complex and interesting challenge! While this idea is certainly not entirely novel, it certainly is a concept that has been only deeply explored in the past four or so years. I’ve personally only seen a handful of content creators online and a select few research papers conducted by graduate students at universities such as Cornell that attempt to implement or explore this idea past a surface level.

I grew up listening to all kinds of Hip-Hop and Rap music thanks to my older brother. Some of these artists include Logic, The Notorious B.I.G., Nas, Kanye West, Hopsin, Eminem, Yelawolf, Big Sean, and a plethora of other rappers. Lately, my roommates and I have been listening to Kanye West quite a lot. This is largely in part to his many big controversies, his meme-like nature, and the recent release of his new studio album Donda 2. I’m not super well versed in all of music, but I do find it incredibly flavorful, catchy, and unique, which is enough motivation for me to pursue this endeavor using his lyrics. Additionally, given his massive discography of 11 studio albums, there is a ton of data to analyze, which should produce interesting results if I can pull this off nicely. Conveniently, I found a compilation of approximately 10,000 lines of lyrics from Kanye West’s discography as of four years ago. This dataset can be found here: <https://github.com/gsurma/text_predictor/tree/master/data/kanye>. This dataset contains lyrics from the following albums: ‘The College Dropout’, ‘808s & Heartbreak’, ‘Yeezus’, ‘Late Registration’, ‘Graduation’, ‘My Beautiful Dark Twisted Fantasy’, ‘Watch the Throne’ and ‘The Life of Pablo’. The way this data is organized is similar to that of Genius lyrics, which includes labels for song verses, hooks, and artist transitions. This is a relatively small dataset, so it is a possibility that I may compile additional lyrics from songs in his most recent albums Donda and Donda 2. Donda 2 is available exclusively only on Kanye’s STEM players, which is a standalone device containing the Donda 2 album amongst other song-mixing capabilities. My roommate purchased one and it could be an interesting endeavor to transcribe some of these lyrics if they’re not available elsewhere. If time does not permit for adding lyrics from Donda 2, I may add lyrics from the album Donda. If I were to incorporate lyrics from these albums, besides self-transcription, I would likely aggregate these lyrics from a reliable lyric website like Genius Lyrics and organize them similar to how the dataset already appears as mentioned above. Again, this is only a possibility and I will assess how likely this is to happen as I dive further into implementation.

The complexity of how humans communicate messages through songs is the main attraction of this project for me. While I can make this project infinitely complex by attempting to analyze the structure of sentences, rhyme patterns, phonetics, slang, and rhyme “flow” amongst the many other aspects of lyric writing, I know time will not entirely permit this. My main focus is not to convey meaningful lyrics nor produce incredible rhyme schemes, but rather to train a model to be able to mimic Kanye West’s vernacular and lyric style. During my research on how I could achieve this goal, I found two main neural structures that were discussed, those being RNNs and LSTMs. Mainly, I plan for the implementation to utilize a LSTM RNN but this may be subject to change. The main attraction of RNNs is the concept of persisting information, and we have seen this concept in action through in-class assignments. I’ve looked into some alternatives of LSTM RNN models and VAEs have shown up multiple times as viable alternatives amongst other deep neural models. Many people online seem to vouch heavily for (auto-regressive) transformer models, which is currently state of the art. Additionally, Gated Recurrent Unit Networks were noted as good alternatives to traditional RNNs. Other suggestions which I found to be less reliable included models like HMMs.

Ultimately, I want this model to be text predictive first and foremost, based on patterns of text which I will hope to define more as my implementation begins. At this point of the planned implementation, I’m not sure if I want to implement a character-level language model or a word-level language model. I believe a character-level model would be more effective at producing truly novel rap, while the word-level model would be a more familiar implementation. Either way, I will be utilizing either PyTorch or TensorFlow for the model implementation and tensors are just super dope generally (after learning how to correctly use them in Assignment 3)! With either the character-level model or word-level model, I will try to batchify the data and organize the implementation similar to that of Assignment 3, since it deals with a similar size dataset and similar machine learning principles. I am anticipating the largest hurdles I will face will be the data preparation and batchification and then model implementation. By the time of the next progress check-in, I am hoping to be done with at least the data preparation and batchification and moving towards beginning the implementation of the model. Judging by some of the other resources regarding text-generation in general, it seems that running tens of thousands, even sometimes hundreds of thousands of epochs is necessary before producing any intelligible output. The running time will certainly be a point of contention, but this is a great opportunity for me to really dive into the Big-O analysis of my own code!

As I mentioned previously, there are a handful of research papers published on this topic of Hip-Hop and Rap lyric generation. The one I found to be the most interesting was “DopeLearning: A Computational Approach to Rap Lyrics Generation” published by Cornell students. The link to the abstract is found here: <https://arxiv.org/abs/1505.04771>. A PDF of the paper is found here: <https://www.kdd.org/kdd2016/papers/files/adf0399-malmiA.pdf>. This paper thoroughly deconstructs rap lyrics down to rhyme schemes, rhyme densities, and song structures to find the best “next-line” of a given song. The idea of rhyme density is pivotal in helping to define features in the dataset, which were ultimately defined to be rhymes, structural similarity between rhymes and semantic similarity between rhymes. The implementation uses a modified K-BOW for representing text, as it considers the K (K=5) previous lines of lyrics. Next, it transforms the data using latent semantic analysis (LSA) to represent each line as a term vector, which is then ranked amongst 100 other lines. Features are then combined using the NN5 feature (neural competition forecasting - sounds very complicated), and the RankSVM method. This implementation of feature extraction seems novel and produces incredible results as an evaluation of the produced lyrics shows that in terms of quantitative rhyme density, the method outperforms the best human rappers by 21%! Ultimately, the researchers here aimed to solve the problem of constructing coherent, rap songs by building one line of text at a time to form a verse. Summarily, this research paper provides a great baseline for how I could further embark on this journey of rap lyric generation, as the scale of this implementation is beyond the scope of time and my expertise! A demo of the lyrics generation algorithm is available at [www.deepbeat.org](http://www.deepbeat.org).

Research articles and sources (note this is just for me and was not attached at the time of submission):

1. <https://arxiv.org/abs/1505.04771>
2. <https://gsurma.medium.com/text-predictor-generating-rap-lyrics-with-recurrent-neural-networks-lstms-c3a1acbbda79#a362>
3. <https://github.com/gsurma/text_predictor/blob/master/data/kanye/input.txt>
4. <https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/>
5. <https://medium.com/deep-learning-with-keras/fundamentals-of-text-generation-745d66238a1f>
6. <https://lilianweng.github.io/posts/2021-01-02-controllable-text-generation/>
7. <https://medium.com/deep-learning-with-keras/part-a2-fundamentals-of-controllable-text-generation-492a3be18ae5>
8. <https://towardsdatascience.com/controlling-text-generation-from-language-models-6334935e80cf>
9. <https://ai.stackexchange.com/questions/25270/are-there-any-good-alternatives-to-an-lstm-language-model-for-text-generation>
10. <https://www.quora.com/What-are-some-alternatives-to-the-LSTM>
11. <https://rodolphe-lampe.medium.com/3-alternatives-to-lstms-1a191a75829a>
12. <https://github.com/carykh/rapLyrics/blob/master/rapSpeaker.py>